

INGENIERÍA MECATRÓNICA



DI_CERO

DIEGO CERVANTES RODRÍGUEZ

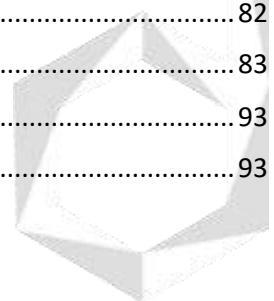
INTELIGENCIA ARTIFICIAL

PYTHON 3.9.7

Asistente Virtual Mark I: LangChain,
Whisper, SpeechRecognition, etc.

Contenido

Whisper: Transcripción de Audio con OpenAI	3
Código Python: Whisper	5
Asistente Virtual - Mark I: Escuchar, Hablar y YouTube	6
Código Python: Asistente Virtual - SpeechRecognition: Reconocer Voz del Micrófono	9
Código Python: Asistente Virtual - Mark I: Hablar y Reproducir Videos en YouTube.....	11
Código Python: Asistente Virtual (A.V.) POO - Mark I: Hablar y Reproducir Videos en YouTube	16
Clase: oidoAsistente.....	17
Clase: vozAsistente	20
Asistente Virtual - Mark I: Hablar, YouTube, OpenAI, LangChain, Alarma, Notas y WhatsApp.....	22
LangChain 🐦 🦜	22
Instalaciones:	28
Código Python: A.V. - Mark I: Hablar, YouTube, OpenAI, Alarma, Notas y WhatsApp.....	34
Clase: oidoAsistente.....	37
Clase: vozAsistente	40
Clase: cerebroLangchainAsistente	41
Clase: alarmaAsistente.....	46
Clase: notasAsistente	48
Clase: whatsappAsistente	50
Asistente Virtual - Mark I: Google Calendar, Mapas, Visión Artificial y Machine Learning	51
API Google Cloud: Herramienta de Google Calendar con Python	51
API MapQuest Developer: Herramienta de Google Calendar con Python	57
OpenCV: Visión Artificial con Python	62
Asistente Virtual - Mark I: Google Calendar, Mapas, Visión Artificial y Machine Learning	65
Clase: oidoAsistente.....	70
Clase: vozAsistente	73
Clase: cerebroLangchainAsistente	74
Clase: alarmaAsistente.....	78
Clase: notasAsistente	81
Clase: whatsappAsistente	82
Clase: googleCalendarAsistente.....	83
Clase: googleCalendar_auth	93
Clase: mapaAsistente	93



Clase: visionArtificialAsistente: Reconocimiento de Colores en Tiempo Real.....	97
Clase: reconocimientoFacialAsistente: Entrenamiento con Machine Learning	101
Función: 1_DatosFaciales.....	106
Función: 2_EntrenamientoFacial	110
Referencias.....	111

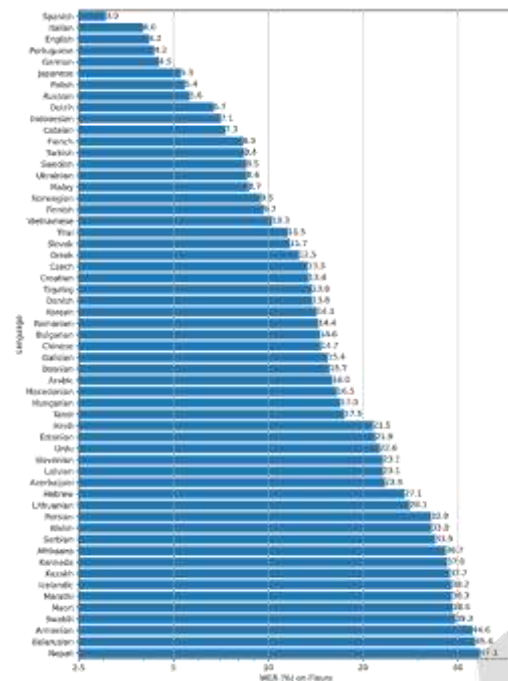
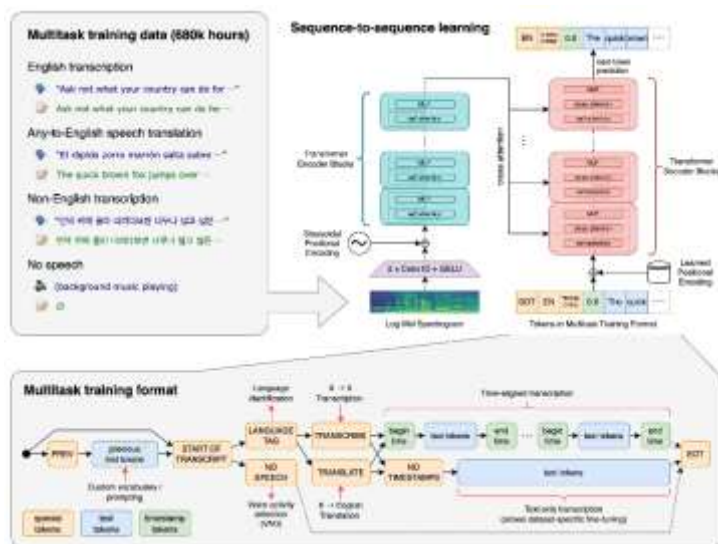


Whisper: Transcripción de Audio con OpenAI

Whisper es un modelo gratuito de inteligencia artificial perteneciente al equipo de OpenAI que se puede descargar de forma local, dando la gran ventaja de que no se debe estar conectado a internet para utilizarlo y puede realizar las siguientes 4 tareas principalmente:

- 1) **Hacer la transcripción de un audio o video hablado en cualquier lenguaje a su texto correspondiente de una forma tan clara que hasta es capaz de detectar signos de puntuación como comas o puntos.**
- 2) **Identificar el lenguaje del audio y texto.**
- 3) **Traducir el texto interpretado del audio a diferentes lenguajes.**
- 4) **Aplicar filtros de ruido y detectar cuando se está hablando y cuando no en el audio, pudiendo incluso detectar las palabras de una canción.**

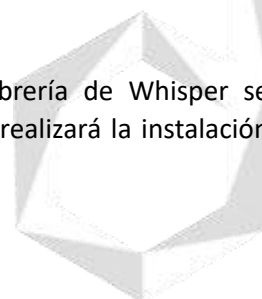
Además, a través de una gráfica llamada Word Error Rate se muestra el error que tiene el modelo al procesar distintos lenguajes, donde 2 de los mejores interpretados son el español e inglés. La herramienta fue entrenada con más de 680,000 horas de distintos audios en distintos lenguajes y se procesa a través de un modelo nombrado transformer, que es un tipo de red neuronal artificial utilizada en el procesamiento del lenguaje natural (NLP).



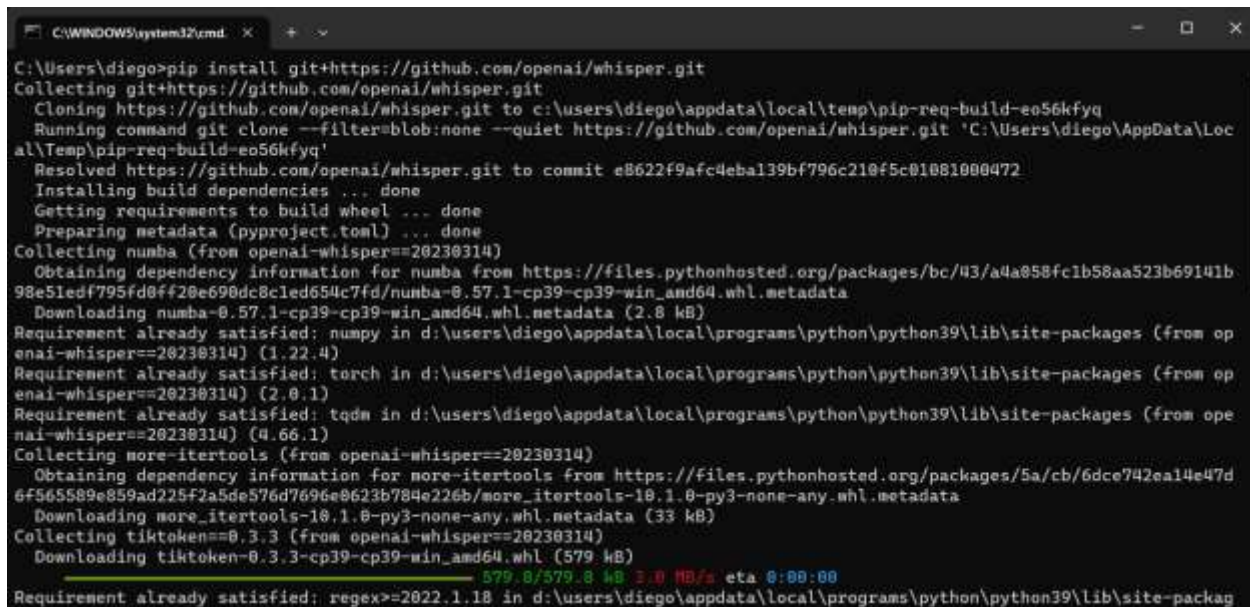
Para realizar la instalación del modelo se deben seguir las instrucciones dadas en el repositorio de Whisper perteneciente al perfil oficial de OpenAI en GitHub, el cual se encuentra en el siguiente enlace:

<https://github.com/openai/whisper>

- **Whisper - OpenAI - Transcripción Audio a Texto:** Para instalar la librería de Whisper se ejecutará el siguiente comando en la consola CMD de Windows, el cual realizará la instalación de la última versión del modelo desde GitHub:



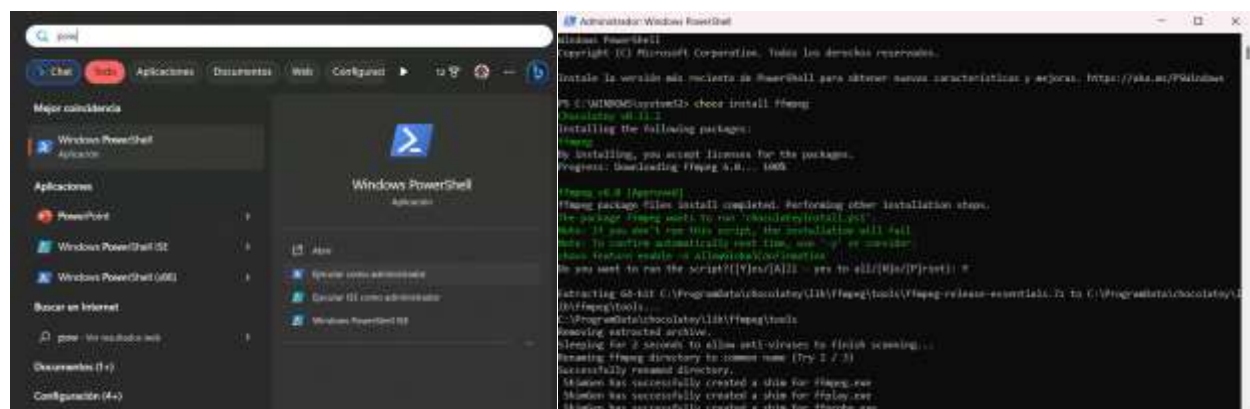
```
pip install git+https://github.com/openai/whisper.git
```



```
C:\Users\diego>pip install git+https://github.com/openai/whisper.git
Collecting git+https://github.com/openai/whisper.git
  Cloning https://github.com/openai/whisper.git to c:\users\diego\appdata\local\temp\pip-req-build-e056kfyq
  Running command git clone --filter=blob:none --quiet https://github.com/openai/whisper.git 'C:\Users\diego\AppData\Local\Temp\pip-req-build-e056kfyq'
  Resolved https://github.com/openai/whisper.git to commit e8622f9afc4eb139bf796c210f5c01081000472
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting numba (from openai-whisper==20230314)
  Obtaining dependency information for numba from https://files.pythonhosted.org/packages/bc/43/a4a858fc1b58aa523b69141b98e51edf795fd0ff20e690dc8c1ed654c7fd/numba-0.57.1-cp39-cp39-win_amd64.whl.metadata
  Downloading numba-0.57.1-cp39-cp39-win_amd64.whl.metadata (2.8 kB)
Requirement already satisfied: numpy in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from openai-whisper==20230314) (1.22.4)
Requirement already satisfied: torch in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from openai-whisper==20230314) (2.0.1)
Requirement already satisfied: tqdm in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from openai-whisper==20230314) (4.66.1)
Collecting more-itertools (from openai-whisper==20230314)
  Obtaining dependency information for more-itertools from https://files.pythonhosted.org/packages/5a/cb/6dce742e14e47d6f565589e859ad225f2a5de576d7696e0623b784e226b/more-itertools-10.1.0-py3-none-any.whl.metadata
  Downloading more_itertools-10.1.0-py3-none-any.whl.metadata (33 kB)
Collecting tiktoken==0.3.3 (from openai-whisper==20230314)
  Downloading tiktoken-0.3.3-cp39-cp39-win_amd64.whl (579 kB)
 579.8/579.8 kB 3.0 MB/s eta 0:00:00
Requirement already satisfied: regex>=2022.1.18 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packag
```

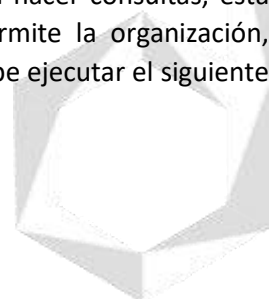
- **ffmpeg - Instalación en Modo Administrador - Transcripción Audio a Texto:** También es importante ejecutar el siguiente comando en la consola CMD de Windows o en la PowerShell para activar las funciones de Whisper, para ello se debe abrir la consola en modo administrador, ya que, si esto no se hace en modo administrador, será identificado por la consola, mostrando un mensaje de advertencia indicando que se podría generar una mala instalación, además, si se usa otro sistema operativo el comando cambia, el mostrado a continuación es para Windows:

```
choco install ffmpeg
```



- **Visualización de Datos - DataFrames - Pandas:** La herramienta llamada **pandas** se puede utilizar de forma opcional si se quiere observar en forma de tablas organizadas los datos devueltos en forma de diccionarios por los modelos al hacer consultas, esta permite integrar un tipo de dato llamado **DataFrame** que permite la organización, análisis y mejor visualización de datos en consola, para ello se debe ejecutar el siguiente comando en la consola CMD de Windows.

```
pip install pandas
```



```
C:\WINDOWS\system32\cmd - X + v
Microsoft Windows [Versión 10.0.22621.2215]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\diego>pip install pandas
Requirement already satisfied: pandas in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (1.3.3)
Requirement already satisfied: numpy>=1.17.3 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pandas) (1.22.4)
Requirement already satisfied: python-dateutil>=2.7.3 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.5 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from python-dateutil>=2.7.3->pandas) (1.12.0)

C:\Users\diego>
```

Código Python: Whisper

```
# -*- coding: utf-8 -*-

#En Python se introducen comentarios de una sola linea con el simbolo #.
#La primera línea de código incluida en este programa se conoce como declaración de codificación o codificación
#de caracteres. Al especificar utf-8 (caracteres Unicode) como la codificación, nos aseguramos de que el archivo
#pueda contener caracteres especiales, letras acentuadas y otros caracteres no ASCII sin problemas, garantizando
#que Python interprete correctamente esos caracteres y evite posibles errores de codificación.
#Se puede detener una ejecución con el comando [CTRL] + C puesto en consola, con el comando "cls" se borra su
#historial y en Visual Studio Code con el botón superior derecho de Play se corre el programa.
#Para comentar en Visual Studio Code varias líneas de código se debe pulsar:
#[CTRL] + K (VSCode queda a la espera). Después pulsa [CTRL] + C para comentar y [CTRL] + U para descomentar.

#IMPORTACIÓN DE LIBRERÍAS:
import whisper #whisper: Librería de OpenAI que permite hacer la transcripción de audio o video a texto.

#whisper.load_model(): Con el método load_model() se pueden cargar los diferentes modelos de lenguaje de la
#librería whisper, si se elige un modelo muy simple, se tendrá un peor desempeño, pero mientras más complejo sea
#el modelo, más se tardará en procesar el audio, a esto dentro de la librería se le llama Parameter y Relative
#Speed, donde a un mayor número de parámetros, se tendrá mayor capacidad de aprendizaje en el modelo y la
#velocidad relativa indica la rapidez con la que el modelo puede generar texto, esta se calcula dividiendo el
#tiempo que tarda el modelo en generar una palabra entre el número de palabras que genera. Las características
#de los modelos disponibles son las siguientes:
# - tiny:   Parameter = 39M,   Memoria que consume = 1GB   y Relative Speed = 32x.
# - base:   Parameter = 74M,   Memoria que consume = 1GB   y Relative Speed = 16x.
# - small:  Parameter = 244M,  Memoria que consume = 2GB   y Relative Speed = 6x.
# - medium: Parameter = 769M,  Memoria que consume = 5GB   y Relative Speed = 2x.
# - large:  Parameter = 1550M, Memoria que consume = 10GB  y Relative Speed = 1x.
#Cuando el programa sea ejecutado por primera vez empezará a descargar el modelo para poderlo utilizar, este
#proceso tarda un poco en terminar, pero después identificará por sí solo el lenguaje y las palabras dichas en
#el audio, no importando si este tenga ruido o si es de alguna canción con instrumentos detrás.
Modelo = whisper.load_model("small")
#whisper.load_model().transcribe(): El método transcribe() sirve para escuchar el audio de un archivo y
```



```

#transcribirlo a texto a través del modelo elegido con el método load_model(). El proceso de carga tarda un poco
#y el tiempo que el modelo se tarda en procesar el audio depende de su duración. Los parámetros que recibe el
#método son la ubicación del archivo de audio que quiere transcribir y el parámetro fp16 que indica si el
#cálculo se realizará en formato de punto flotante de 16 bits (FP16), pero como en algunas CPUs no se admite
#esta funcionalidad, se coloca como False, por lo que se está utilizando punto flotante de 32 bits (FP32) en su
#lugar. Además cabe mencionar que el audio escuchado por el método será recortado en cachos de 30 segundos y
#estos cachos serán igualmente divididos en los pedazos que los conforman.

TranscripcionAudio_a_Texto = Modelo.transcribe("C:/Users/diego/OneDrive/Documents/Aprendiendo/Python/5.-Inteligencia
Artificial/0.-Archivos_Ejercicios_Python/Till I Collapse - Eminem.mp3", fp16 = False)

#print(): Método para imprimir un mensaje en consola y después dar un salto de línea (Enter).
print("\n\nDiccionario generado por Whisper:\n" + str(TranscripcionAudio_a_Texto) + "\n\n")
print("Texto traducido por Whisper de un audio de música:\n" + str(TranscripcionAudio_a_Texto["text"]) + "\n")
print("El lenguaje del audio es:\n" + str(TranscripcionAudio_a_Texto["language"]) + "\n\n")
print("Texto separado en cachos dentro de un segmento de 30 segundos del audio:\n" + str(TranscripcionAudio_a_Texto["segments"])
+ "\n\n")

#DataFrame: Un DataFrame es una estructura de datos de dos dimensiones que se utiliza para almacenar datos en
#forma de tablas donde las filas representan las observaciones de los datos, y las columnas representan las
#variables de los datos de manera similar a las hojas de cálculo de Excel. En este caso se podría utilizar
#para observar en consola de mejor manera los pedazos separados dentro de los 30 segundos de audio.

import pandas as pd

#pd.DataFrame(): El constructor del objeto DataFrame() permite desplegar en forma de tabla los datos de un
#diccionario que reciba como parámetro, pudiendo después indicar a través de una lista anidada que esté
#después del método, exactamente cuales son los keys que quiero observar en la tabla.

DataFrameAudio = pd.DataFrame(TranscripcionAudio_a_Texto["segments"])[['start', 'end', 'text']]

print("DataFrame del texto separado en cachos:\n" + str(DataFrameAudio) + "\n\n")

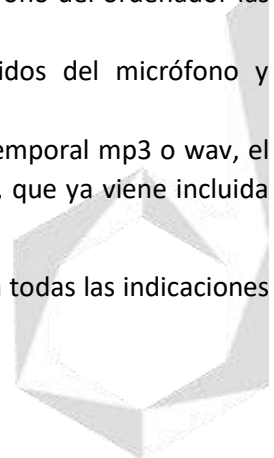
```

Asistente Virtual - Mark I: Escuchar, Hablar y YouTube

Como la librería Whisper de OpenAI que hace el reconocimiento de audio y lo transforma a texto para que el programa la interprete no se puede utilizar en tiempo real, sino que analiza pedazos de audio que ya hayan sido grabados y salvados, se utilizarán las siguientes librerías de apoyo para escuchar las instrucciones del usuario:

- **SpeechRecognition:** Librería utilizada para recopilar por medio del micrófono del ordenador las instrucciones que indique el usuario.
- **pydub:** Biblioteca que sirve para procesar los datos de audio recibidos del micrófono y guardarlos en un archivo mp3 o wav.
- **Tempfile:** El audio recopilado por el usuario se guardará en un archivo temporal mp3 o wav, el cual será almacenado en la carpeta temp a través de la librería tempfile, que ya viene incluida por default con Python

Esta acción se estará repitiendo cada cierto tiempo para captar de forma correcta todas las indicaciones del usuario.





- **SpeechRecognition - Oído Asistente Virtual:** La habilidad de utilizar el micrófono del ordenador para reconocer nuestra voz a través de Python y así poderle pasar instrucciones al modelo de Whisper se puede proporcionar al ejecutar el siguiente comando en la consola CMD de Windows.

```
pip install SpeechRecognition
```

```
C:\WINDOWS\system32\cmd. X + -
C:\Users\diego>pip install SpeechRecognition
Requirement already satisfied: SpeechRecognition in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (3.8.1)
C:\Users\diego>
```

- **pydub - Oído Asistente Virtual:** Una vez recabado el audio del micrófono, para procesar esos datos y guardarlos en un archivo mp3 o wav se debe ejecutar el siguiente comando en la consola CMD de Windows, permitiéndonos así utilizar la librería pydub.

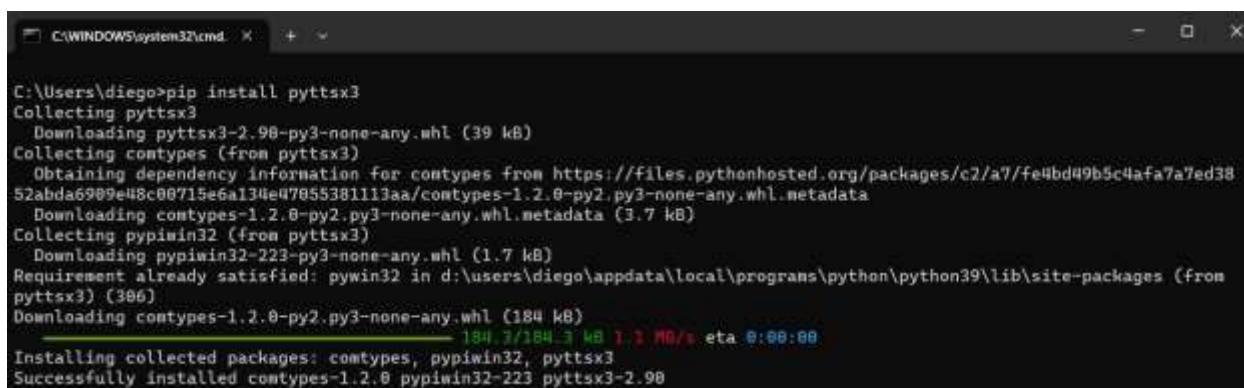
```
pip install pydub
```

```
C:\WINDOWS\system32\cmd. X + -
C:\Users\diego>pip install pydub
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub
Successfully installed pydub-0.25.1
C:\Users\diego>
```


Ya que se hayan instalado las librerías que le permiten **escuchar al asistente virtual** y habiendo procesado esa información de audio para convertirla en un texto, se deben instalar las librerías que le permitan **hablar y responder las peticiones realizadas por el usuario**, en un inicio se utilizarán las voces integradas en el sistema operativo de Windows, ya que si se busca utilizar una voz clonada con inteligencia artificial por medio de las herramientas de Eleven Labs, FakeYou, etc. se debe realizar un pago adicional:

- **pyttsx3 - Voz del Asistente Virtual - Incluida en Windows:** Para dar la habilidad de hablar al asistente virtual se deben instalar 2 librerías: pyttsx3 y pyaudio, para ello se deben ejecutar los siguientes comandos en la consola CMD de Windows.

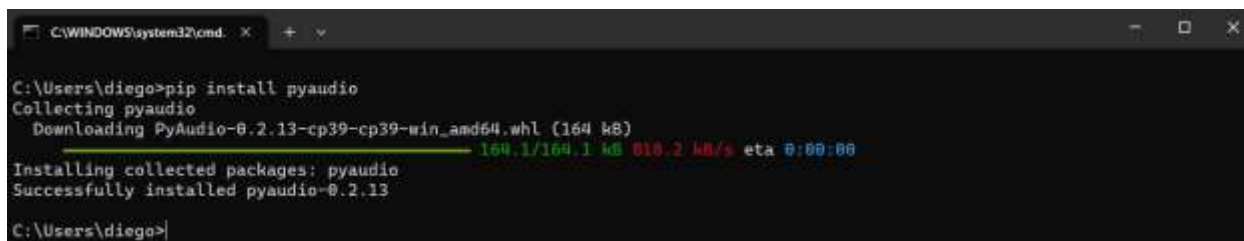
```
pip install pyttsx3
```



```
C:\WINDOWS\system32\cmd. X + v
C:\Users\diego>pip install pyttsx3
Collecting pyttsx3
  Downloading pyttsx3-2.90-py3-none-any.whl (39 kB)
Collecting ctypes (from pyttsx3)
  Obtaining dependency information for ctypes from https://files.pythonhosted.org/packages/c2/a7/fe4bd49b5c4afa7a7ed3852abda6909e48c00715e6a134e47855381113aa/ctypes-1.2.0-py2.py3-none-any.whl.metadata
  Downloading ctypes-1.2.0-py2.py3-none-any.whl.metadata (3.7 kB)
Collecting pypiwin32 (from pyttsx3)
  Downloading pypiwin32-223-py3-none-any.whl (1.7 kB)
Requirement already satisfied: pywin32 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyttsx3) (306)
Downloading ctypes-1.2.0-py2.py3-none-any.whl (184 kB)
 184.7/184.3 kB 1.1 MB/s eta 0:00:00
Installing collected packages: ctypes, pypiwin32, pyttsx3
Successfully installed ctypes-1.2.0 pypiwin32-223 pyttsx3-2.90
```

- **pyaudio - Voz del Asistente Virtual:** La librería pyaudio aporta a la instalación de la librería pyttsx3 para dar la habilidad de reproducir un audio en nuestra computadora a través de Python y así darle voz al asistente virtual.

```
pip install pyaudio
```

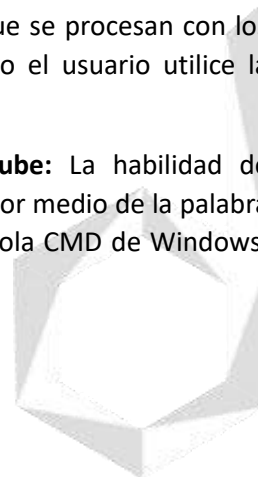


```
C:\WINDOWS\system32\cmd. X + v
C:\Users\diego>pip install pyaudio
Collecting pyaudio
  Downloading PyAudio-0.2.13-cp39-cp39-win_amd64.whl (164 kB)
 164.1/164.1 kB 0.2 MB/s eta 0:00:00
Installing collected packages: pyaudio
Successfully installed pyaudio-0.2.13
C:\Users\diego>
```

Finalmente, como en el Mark1 inicialmente las funciones que se incluirán serán las de mimetizar las instrucciones escuchadas del usuario, para así comprobar la velocidad con las que se procesan con los distintos modelos de Whisper y la de reproducir un video en YouTube cuando el usuario utilice la palabra “reproducir”, se realizará la instalación de una librería adicional.

- **pywhatkit - Acción Asistente Virtual - Reproducir Videos en YouTube:** La habilidad de reproducir un video en YouTube cuando se le ordene al asistente virtual por medio de la palabra “reproduce” se proporciona al ejecutar el siguiente comando en la consola CMD de Windows, que realiza la instalación de la librería pywhatkit.

```
pip install pywhatkit
```



```
C:\WINDOWS\system32\cmd - + v
C:\Users\diego>pip install pywhatkit
Requirement already satisfied: pywhatkit in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (5.4)
Requirement already satisfied: Pillow in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pywhatkit) (10.0.0)
Requirement already satisfied: pyautogui in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pywhatkit) (0.9.54)
Requirement already satisfied: requests in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pywhatkit) (2.31.0)
Requirement already satisfied: wikipedia in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pywhatkit) (1.4.0)
Requirement already satisfied: Flask in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pywhatkit) (2.3.3)
Requirement already satisfied: Werkzeug>=2.3.7 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from Flask->pywhatkit) (2.3.7)
Requirement already satisfied: Jinja2>=3.1.2 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from Flask->pywhatkit) (3.1.2)
Requirement already satisfied: itsdangerous>=2.1.2 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from Flask->pywhatkit) (2.1.2)
Requirement already satisfied: click>=8.1.3 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (
```

Código Python: Asistente Virtual - SpeechRecognition: Reconocer Voz del Micrófono

Al realizar las pruebas con este código se debe comprobar que el sistema de audio se conecte bien con la librería SpeechRecognition, para ello se puede acceder a la opción de Configuración → Sistema → Sonido → Salida o Entrada → Dispositivo → Configuración de entrada o de salida → Probar.

```
# -*- coding: utf-8 -*-

#En Python se introducen comentarios de una sola linea con el simbolo #.
#La primera línea de código incluida en este programa se conoce como declaración de codificación o codificación
#de caracteres. Al especificar utf-8 (caracteres Unicode) como la codificación, nos aseguramos de que el archivo
#pueda contener caracteres especiales, letras acentuadas y otros caracteres no ASCII sin problemas, garantizando
#que Python interprete correctamente esos caracteres y evite posibles errores de codificación.
#Se puede detener una ejecución con el comando [CTRL] + C puesto en consola, con el comando "cls" se borra su
#historial y en Visual Studio Code con el botón superior derecho de Play se corre el programa.
#Para comentar en Visual Studio Code varias líneas de código se debe pulsar:
#[CTRL] + K (VSCode queda a la espera). Después pulsa [CTRL] + C para comentar y [CTRL] + U para descomentar.
import speech_recognition #SpeechRecognition: Librería para recabar audio del micrófono en tiempo real.
import time               #time: Librería de manejo de tiempos, como retardos, contadores, etc.

#speech_recognition.Recognizer(): El objeto Recognizer() se utiliza para crear un reconocedor de voz, el cual
#puede utilizarse para reconocer instrucciones dadas al asistente virtual por medio del habla humana.
listenerAudio = speech_recognition.Recognizer()
print("Inicializando listener con la librería speech_recognition")

#Bucle indeterminado while para ejecutar siempre este programa que reconozca el audio del usuario hasta que
#diga bye Timmy.
while True:
    #MANEJO DE EXCEPCIONES: Es una parte de código que se conforma de dos partes, try y except:
    # - Primero se ejecuta el código que haya dentro del try y si es que llegara a ocurrir una excepción durante
    # su ejecución, el programa brinca al código del except.
    # - En la parte de código donde se encuentra la palabra reservada except, se ejecuta cierta acción cuando
```



```

# ocurra el error esperado.

#Se utiliza esta arquitectura de código cuando se quiera efectuar una acción donde se espera que pueda
#ocurrir un error durante su ejecución.

try:

    #with as source: La instrucción with se utiliza para definir una variable que tiene asignada un método
    #o recurso específico, el cual puede ser un archivo, una conexión a una base de datos, la cámara,
    #micrófono o cualquier otro objeto que requiera ser cerrado. La palabra clave as se utiliza para asignar
    #dicho recurso a una variable que puede utilizarse para acceder al recurso dentro del bloque with.
    #Dentro del manejo de excepciones la instrucción with as source: asegura que se cierre el recurso,
    #incluso si se produce un error dentro del bloque try.
    #speech_recognition.Microphone(): El objeto Microphone puede utilizarse para grabar audio.
    with speech_recognition.Microphone() as microfono:

        #speech_recognition.Recognizer().adjust_for_ambient_noise(): El método adjust_for_ambient_noise()
        #sirve para aplicar un filtro a la señal de sonido que quite el ruido de fondo recibido en su
        #parámetro.
        listenerAudio.adjust_for_ambient_noise(microfono, duration = 0.2)
        print("Quitando ruido de fondo... ya puedes hablar.")

        #time.time(): El método time() se utiliza para devolver un valor de tipo flotante que representa la
        #cantidad de segundos transcurridos desde el 1 de enero de 1970 a las 00:00:00 horas (UTC) hasta el
        #momento actual donde se ejecute este método, pero eso no es tan importante, sino que con este se
        #pueden medir intervalos de tiempo al declararlo dos veces y realizar una resta.
        tiempoInicio = time.time()

        #speech_recognition.Recognizer().listen(): El método listen() sirve para poder escuchar de una
        #fuente de audio en tiempo real y convertirlo a texto, para ello primero se tuvo que haber
        #instanciado la clase Recognizer.
        audioMic = listenerAudio.listen(microfono, timeout = 5, phrase_time_limit = 5)
        print("Microfono encendido y escuchando...")

        tiempoFinal = time.time()
        intervaloTiempoQuitarRuido = tiempoInicio - tiempoFinal

        print("El tiempo que se tardó en configurarse el método adjust_for_ambient_noise() fue de: " +
str(intervaloTiempoQuitarRuido))

        #speech_recognition.Recognizer().recognize_google(): El método recognize_google() utiliza el
        #servicio de Google para convertir el audio en texto, la desventaja de usar este es que no se corre
        #de manera local, a fuerza debemos estar conectados a internet para que esto se ejecute, este recibe
        #como parámetro un objeto de la librería speech_recognition al que se le haya aplicado la función
        #listen().
        texto = listenerAudio.recognize_google(audioMic)
        print("Traduciendo microfono a texto... Texto reconocido:\n\t" + str(texto))

        #Si se reconoce que el usuario dice bye Timmy, se rompe el bucle while y se termina la ejecución del
        #programa.
        if (texto == "bye Timmy"):
            break

#Para identificar el tipo de excepción que ha ocurrido y utilizarlo en la instrucción except, se puede
#utilizar la clase Exception, que es una clase incorporada en Python utilizada para describir todos los
#tipos de excepciones, luego de colocar el nombre de la clase Exception se usa la palabra reservada "as"

```



```

#seguida de un nombre de variable, esto nos permitirá acceder a la instancia de la excepción y utilizarla
#dentro del except.
except Exception as error:
    #Si ocurre un error, se imprime en consola el mensaje de Podrías repetir lo que dijiste? y además se
    #vuelve a crear el objeto speech_recognition.Recognizer() para volver a intentar escuchar lo que el
    #usuario dijo.
    listenerAudio = speech_recognition.Recognizer()
    print("Podrías repetir lo que dijiste? Ocurrió un error: " + str(error))
    continue

```

Código Python: Asistente Virtual - Mark I: Hablar y Reproducir Videos en YouTube

```

# -*- coding: utf-8 -*-

#En Python se introducen comentarios de una sola linea con el simbolo #.
#La primera línea de código incluida en este programa se conoce como declaración de codificación o codificación
#de caracteres. Al especificar utf-8 (caracteres Unicode) como la codificación, nos aseguramos de que el archivo
#pueda contener caracteres especiales, letras acentuadas y otros caracteres no ASCII sin problemas, garantizando
#que Python interprete correctamente esos caracteres y evite posibles errores de codificación.
#Se puede detener una ejecución con el comando [CTRL] + C puesto en consola, con el comando "cls" se borra su
#historial y en Visual Studio Code con el botón superior derecho de Play se corre el programa.
#Para comentar en Visual Studio Code varias líneas de código se debe pulsar:
#[CTRL] + K (VSCode queda a la espera). Después pulsa [CTRL] + C para comentar y [CTRL] + U para descomentar.

#IMPORTACIÓN DE LIBRERÍAS:
import io          #io: Librería que permite la manipulación de los archivos y carpetas de nuestro ordenador.
#AudioSegment: Clase de la librería pydub que permite tomar el audio que perciba el micrófono del ordenador y
#luego que eso se pueda convertir en un archivo mp3 o wav.
import pydub       #pydub: Librería para procesar y exportar a un archivo datos de audio.
import speech_recognition as sr #sr: Librería que permite recabar audio del micrófono en tiempo real.
import whisper     #whisper: Librería de OpenAI que permite hacer la transcripción de audio o video a texto.
import tempfile    #tempfile: Librería que crea y maneja archivos temporales.
import os          #os: Librería que permite acceder a funciones y métodos relacionados con el sistema operativo.
import pyttsx3     #pyttsx3: Biblioteca que sirve para generar una voz disponible en el sistema operativo.
#pywhatkit: Pywhatkit es una biblioteca de Python que permite enviar mensajes de texto a través de WhatsApp
#incluso si no estás en su lista de contactos, realizar búsquedas en la web, reproducir canciones en YouTube,
#realizar búsquedas en Wikipedia, ejecutar comandos en consola, etc.
import pywhatkit

#CREACIÓN DE ARCHIVO TEMPORAL:
#tempfile.mkdtemp(): El método mkdtemp() crea un directorio temporal en la carpeta predeterminada del sistema
#operativo, la cual es llamada temp y es una ubicación en el sistema operativo donde se almacenan archivos
#temporales generados por diversas aplicaciones y procesos.
archivoTemporal = tempfile.mkdtemp()
#os.path.join(): El método join() se utiliza para unir varios componentes de ruta (variables y constantes) en

```



```

#una sola ruta. Esto se utilizará para crear el archivo wav o mp3 que guardará un pedazo de la instrucción del
#usuario. Cabe mencionar que WAV es un formato de archivo de audio sin pérdidas, lo que significa que todos los
#datos de audio originales se conservan. Esto hace que los archivos WAV sean de mayor calidad que los archivos
#MP3, pero también los hace más grandes y pesados.

rutaArchivo = os.path.join(archivoTemporal, 'audioTemporal.wav')
print("Esta es la ruta del archivo temporal de las instrucciones recibidas en audio:\n" + str(rutaArchivo))

#RECONOCIMIENTO DE LAS INSTRUCCIONES DADAS AL USUARIO:
#speech_recognition.Recognizer(): El objeto Recognizer() se utiliza para crear un reconocedor de voz, el cual
#puede utilizarse para reconocer instrucciones dadas al asistente virtual por medio del habla humana.
listenerInstrucciones = sr.Recognizer()

#VOZ UTILIZADA POR EL ASISTENTE VIRTUAL:
#pyttsx3.init(): El método pyttsx3.init() se utiliza para inicializar el motor de texto a voz, el cual dará la
#habilidad de hablar al asistente virtual, convirtiendo el texto retornado por el modelo de Whisper en audio que
#salga de la computadora.
motorVoz = pyttsx3.init()

#pyttsx3.init().getProperty("voices"): El método getProperty() se utiliza para obtener una lista de las voces
#disponibles en el motor de texto a voz incluidos en el sistema operativo del ordenador.
voz_AsistenteVirtual = motorVoz.getProperty("voices")

#Bucle for para mostrar todas las opciones de sintetizadores de voz disponibles en el sistema operativo, el
#número de opciones dependerá de los lenguajes que pueda manejar el sistema operativo de la computadora que
#digan text to speech y se pueden ver al ingresar a la opción de: Windows -> Configuración -> Hora e Idioma ->
#Idioma y Región -> Idioma -> Idiomas que diga texto a voz. Si se quiere se podría agregar más idiomas.
for i in range(len(voz_AsistenteVirtual)):
    print(voz_AsistenteVirtual[i])

#pyttsx3.init().setProperty(): Este método toma dos parámetros y no devuelve ningún valor porque indica las
#siguientes características del sintetizador de voz del asistente virtual:
# - name: El nombre de la propiedad que se desea establecer.
# - rate: Indica la velocidad de la voz en palabras por minuto.
# - volume: El volumen de la voz, de 0 a 1.
# - voices: Lista de todas las voces disponibles.
# - voice: La voz que se utilizará para hablar.
# - language: El idioma que se utilizará para hablar.
# - engine: Controlador de voz que se utilizará.
# - debug: Variable booleana que indica si se debe habilitar el modo de depuración o no.
# - value: El valor de la propiedad que se desea establecer.
motorVoz.setProperty("rate", 145)
motorVoz.setProperty("voice", voz_AsistenteVirtual[3].id)

#hablar(textoAsistente): Función propia que permite al asistente virtual hablar por medio de la bocina del
#ordenador.
def hablar(textoAsistente):
    #pyttsx3.init().say(): El método say() se utiliza para convertir texto en audio y emitirlo a través del
    #altavoz del sistema.

```



```

motorVoz.say(textoAsistente)

#pytttsx3.init().runAndWait(): El método runAndWait() bloquea la ejecución del programa hasta que el motor
#de texto a voz haya terminado de hablar.

motorVoz.runAndWait()

#escuchaInstrucciones(textoAsistente): Función propia que se encarga de escuchar y reconocer el audio por medio
#del micrófono del ordenador las palabras dichas por el usuario, esto se tarda un poco en cargar la primera vez
#que se corre el programa debido a que el método speech_recognition.Recognizer().listen() necesita un tiempo para
#inicializarse y configurarse; si se está utilizando una computadora con poca potencia o micrófonos de baja
#calidad, es posible que se necesite esperar más tiempo.

def escuchaInstrucciones():
    #MANEJO DE EXCEPCIONES: Es una parte de código que se conforma de dos partes, try y except:
    # - Primero se ejecuta el código que haya dentro del try y si es que llegara a ocurrir una excepción durante
    # su ejecución, el programa brinca al código del except.
    # - En la parte de código donde se encuentra la palabra reservada except, se ejecuta cierta acción cuando
    # ocurra el error esperado.
    #Se utiliza esta arquitectura de código cuando se quiera efectuar una acción donde se espera que pueda
    #ocurrir un error durante su ejecución.
    try:
        #with as source: La instrucción with se utiliza para definir una variable que tiene asignada un método
        #o recurso específico, el cual puede ser un archivo, una conexión a una base de datos, la cámara,
        #micrófono o cualquier otro objeto que requiera ser cerrado. La palabra clave as se utiliza para asignar
        #dicho recurso a una variable que puede utilizarse para acceder al recurso dentro del bloque with.
        #Dentro del manejo de excepciones la instrucción with as source: asegura que se cierre el recurso,
        #incluso si se produce un error dentro del bloque try.
        #speech_recognition.Microphone(): El objeto Microphone puede utilizarse para grabar audio.
        with sr.Microphone() as microfono:
            print("\n\n-----Hola soy T.I.M.M.Y. Mark I tu asistente virtual, en que te puedo ayudar?...-----")
            #speech_recognition.Recognizer().adjust_for_ambient_noise(): El método adjust_for_ambient_noise()
            #sirve para aplicar un filtro a la señal de sonido que quite el ruido de fondo recibido en su
            #parámetro.
            listenerInstrucciones.adjust_for_ambient_noise(microfono)
            print("Quitando ruido de fondo... ya puedes hablar.")
            #speech_recognition.Recognizer().listen(): El método listen() sirve para poder escuchar de una
            #fuente de audio en tiempo real y convertirlo a texto, para ello primero se tuvo que haber
            #instanciado la clase Recognizer.
            textoInstruccionUsuario = listenerInstrucciones.listen(microfono)
            print("Microfono encendido y escuchando...")
            #io.BytesIO(): Método que sirve para crear un flujo de bytes en memoria que puede utilizarse para
            #almacenar datos binarios, como imágenes, audio y video.
            #speech_recognition.Recognizer().listen().get_wav_data(): El método get_wav_data() se utiliza para
            #almacenar el texto recibido en el micrófono del ordenador en un flujo de datos binarios que se
            #puedan guardar en un archivo de audio con extensión wav.
            datosAudio = io.BytesIO(textoInstruccionUsuario.get_wav_data())
            #pydub.AudioSegment().from_file(): El método from_file() perteneciente a la clase AudioSegment se

```




```

#utiliza para cargar solamente los datos de audio donde se escuche la voz del usuario, logrando así
#que no se haga nada cuando el usuario está en silencio.
archivoAudio = pydub.AudioSegment.from_file(datosAudio)

print("Mandando texto del microfono a un archivo de audio wav...")

#pydub.AudioSegment().from_file().export(path, file_type): Lo que hace el método export() es
#exportar un segmento de audio a un archivo.

archivoAudio.export(rutaArchivo, format = 'wav')

print("Texto del microfono guardado en un archivo de audio temporal tipo wav...")

#Para identificar el tipo de excepción que ha ocurrido y utilizarlo en la instrucción except, se puede
#utilizar la clase Exception, que es una clase incorporada en Python utilizada para describir todos los
#tipos de excepciones, luego de colocar el nombre de la clase Exception se usa la palabra reservada "as"
#seguida de un nombre de variable, esto nos permitirá acceder a la instancia de la excepción y utilizarla
#dentro del except.
except Exception as error:

    print(error)

#Se retorna la ruta del archivo de la función porque de ahí extraerá el archivo de audio el modelo Whisper
#para procesarlo.

return rutaArchivo

#transcripcionWhisper(): Función propia que se encarga de recibir el archivo de audio recortado que contiene
#la instrucción mandada al asistente virtual para pasársela al modelo de Whisper para que la transcriba a texto.
def transcripcionWhisper(audioRecortado):

    #whisper.load_model(): Con el método load_model() se pueden cargar los diferentes modelos de lenguaje de la
    #librería whisper, si se elige un modelo muy simple, se tendrá un peor desempeño, pero mientras más complejo
    #sea el modelo, más se tardará en procesar el audio, a esto dentro de la librería se le llama Parameter y
    #Relative Speed, donde a un mayor número de parámetros, se tendrá mayor capacidad de aprendizaje en el
    #modelo y la velocidad relativa indica la rapidez con la que el modelo puede generar texto, esta se calcula
    #dividiendo el tiempo que tarda el modelo en generar una palabra entre el número de palabras que genera. Las
    #características de los modelos disponibles son las siguientes:

    # - tiny:   Parameter = 39M,   Memoria que consume = 1GB   y Relative Speed = 32x.
    # - base:   Parameter = 74M,   Memoria que consume = 1GB   y Relative Speed = 16x.
    # - small:  Parameter = 244M,  Memoria que consume = 2GB   y Relative Speed = 6x.
    # - medium: Parameter = 769M,  Memoria que consume = 5GB   y Relative Speed = 2x.
    # - large:  Parameter = 1550M, Memoria que consume = 10GB  y Relative Speed = 1x.

    #Cuando el programa sea ejecutado por primera vez empezará a descargar el modelo para poderlo utilizar, este
    #proceso tarda un poco en terminar, pero después identificará por sí solo el lenguaje y las palabras dichas
    #en el audio, no importando si este tenga ruido o si es de alguna canción con instrumentos detrás.

    Modelo = whisper.load_model("small")

    #whisper.load_model().transcribe(): El método transcribe() sirve para escuchar el audio de un archivo y
    #transcribirlo a texto a través del modelo elegido con el método load_model(). El proceso de carga tarda un
    #poco y el tiempo que el modelo se tarda en procesar el audio depende de su duración. Los parámetros que
    #recibe el método son la ubicación del archivo de audio que quiere transcribir y el parámetro fp16 que
    #indica si el cálculo se realizará en formato de punto flotante de 16 bits (FP16), pero como en algunas CPUs
    #no se admite esta funcionalidad, se coloca como False, por lo que se está utilizando punto flotante de 32
    #bits (FP32) en su lugar. Además cabe mencionar que el audio escuchado por el método será recortado en

```



```

#cachos de 30 segundos y estos serán igualmente divididos en pedazos.

TranscripcionAudio_a_Texto = Modelo.transcribe(audioRecortado, language = "spanish", fp16 = False)
print("Interpretando texto guardado en un archivo temporal de audio wav con el modelo Whisper...\t")
#whisper.load_model().transcribe()["text"]: Texto traducido por Whisper de un archivo de audio.
return TranscripcionAudio_a_Texto["text"]

#__name__ == __main__: Método main, esta función es super importante ya que sirve para instanciar las clases del
#programa y ejecutar sus métodos, en python pueden existir varios métodos main en un solo programa, aunque no es
#una buena práctica.
if (__name__ == "__main__"):
    hablar("Hola soy TIMI Mark 1, tu asistente virtual, puedo mimetizar tu voz y reproducir canciones en YoTube, en que te
    puedo ayudar?")
    while True:
        try:
            respuestaAsistenteVirtual = transcripcionWhisper(escuchaInstrucciones())
            hablar(respuestaAsistenteVirtual)
            print(respuestaAsistenteVirtual)
            #Si se reconoce que el usuario dice bye Timmy, se rompe el bucle while y se termina la ejecución del
            #programa.
            if (respuestaAsistenteVirtual == " Adiós, Timmy." or respuestaAsistenteVirtual == " Bye, Timmy." or
            respuestaAsistenteVirtual == " Adiós, tímí." or respuestaAsistenteVirtual == " Adiós, Timí." or respuestaAsistenteVirtual ==
            " Adiós, teamy."):
                hablar("Bye di0, un gusto haberte ayudado..")
                print("-----Bye di_cer0, un gusto haberte ayudado..-----")
                break
            #Si se reconoce que el usuario dice la palabra reproduce, el asistente virtual por medio de la librería
            #pywhatkit reproduce el video que el usuario dijo en YouTube.
            elif (("reproduce" or "Reproduce") in respuestaAsistenteVirtual):
                #.replace().lower(): Lo que hace el método replace() es reemplazar todas las palabras que
                #aparezcan en un string en otra cadena específica y el método lower() sirve para convertir todas
                #las letras de una cadena en minúsculas, esto es importante hacerlo porque al reproducir una
                #canción o mandar una búsqueda a internet, siempre es mejor tener puras minúsculas.
                cancionYoutube = respuestaAsistenteVirtual.replace('reproduce', '').lower()
                hablar("Ok di0, voy a reproducir una canción del artista que pediste en YouTube")
                #pywhatkit.playonyt(): El método playonyt() de la librería pywhatkit permite reproducir un
                #video de YouTube en el navegador web predeterminado del sistema.
                pywhatkit.playonyt(cancionYoutube)
                print("-----Ok di_cer0, ya reproducí la canción del artista que pediste en YouTube.-----")
            except Exception as errorEjecucion:
                hablar("Lo siento, no escuché bien lo que dijiste debido a este error...")
                print(errorEjecucion)
                continue
#El límite de la duración de audio que puede procesar T.I.M.M.Y. en las instrucciones que se le manda es de 15
#segundos y el Mark I en sus respuestas tiene un delay de:
# - tiny: Delay de 2 segundos cuando le mando un comando hablado usando el comando.

```



```
# - base: Delay de 5 segundos cuando le mando un comando hablado usando el comando.  
# - small: Delay de 10 segundos cuando le mando un comando hablado usando el comando.
```

Código Python: Asistente Virtual (A.V.) POO - Mark I: Hablar y Reproducir Videos en YouTube

```
# -*- coding: utf-8 -*-  
  
#En Python se introducen comentarios de una sola linea con el simbolo #.  
#La primera línea de código incluida en este programa se conoce como declaración de codificación o codificación  
#de caracteres. Al especificar utf-8 (caracteres Unicode) como la codificación, nos aseguramos de que el archivo  
#pueda contener caracteres especiales, letras acentuadas y otros caracteres no ASCII sin problemas, garantizando  
#que Python interprete correctamente esos caracteres y evite posibles errores de codificación.  
#Se puede detener una ejecución con el comando [CTRL] + C puesto en consola, con el comando "cls" se borra su  
#historial y en Visual Studio Code con el botón superior derecho de Play se corre el programa.  
#Para comentar en Visual Studio Code varias líneas de código se debe pulsar:  
#[CTRL] + K (VSCode queda a la espera). Después pulsa [CTRL] + C para comentar y [CTRL] + U para descomentar.  
  
#IMPORTACIÓN DE LIBRERÍAS:  
#pywhatkit: Pywhatkit es una biblioteca de Python que permite enviar mensajes de texto a través de WhatsApp  
#incluso si no estás en su lista de contactos, realizar búsquedas en la web, reproducir canciones en YouTube,  
#realizar búsquedas en Wikipedia, ejecutar comandos en consola, etc.  
import pywhatkit  
  
#IMPORTACIÓN DE CLASES: Cuando se quiera importar una clase, el nombre de esta no puede empezar con un número,  
#sino cuando la quiera importar obtendré un error y se va accediendo a las carpetas o también llamados paquetes  
#en la programación orientada a objetos (POO), por medio de puntos:  
# - Directorio normal:      carpeta1/carpeta2/carpeta3  
# - Directorio paquetes:   carpeta1.carpeta2.carpeta3  
#La parte del directorio se coloca después de la palabra reservada from y la clase a importar después de import.  
from POO_AsistenteVirtualMarkI.oidoAsistente import EscucharMicrofono  
from POO_AsistenteVirtualMarkI.vozAsistente import vozWindows  
  
#__name__ == __main__: Método main, esta función es super importante ya que sirve para instanciar las clases del  
#programa y ejecutar sus métodos, en python pueden existir varios métodos main en un solo programa, aunque no es  
#una buena práctica.  
if (__name__ == "__main__"):  
    oidoAsistenteVirtual = EscucharMicrofono() #Instancia de la clase propia EscucharMicrofono.  
    vozAsistenteVirtual = vozWindows() #Instancia de la clase propia vozWindows.  
    vozAsistenteVirtual.hablar("Hola soy TIMMY Mark 1, tu asistente virtual, puedo mimetizar tu voz y reproducir canciones  
en YoTube, en que te puedo ayudar?")  
    while True:  
        try:  
            #EscucharMicrofono.oidoAsistenteVirtual(): Método propio de la clase EscucharMicrofono que se  
            #encarga de escuchar lo que dice el usuario, almacenar eso en un archivo temporal y luego pasárselo  
            #al modelo Whisper para que lo transcriba a texto.
```



```

    respuestaAsistenteVirtual = oidoAsistenteVirtual.oidoAsistenteVirtual()
    vozAsistenteVirtual.hablar(respuestaAsistenteVirtual)
    print(respuestaAsistenteVirtual)

    #Si se reconoce que el usuario dice bye Timmy, se rompe el bucle while y se termina la ejecución del
    #programa.

    if (respuestaAsistenteVirtual == " adiós, timmy." or respuestaAsistenteVirtual == " bye, timmy." or
respuestaAsistenteVirtual == " adiós, timí." or respuestaAsistenteVirtual == " adiós, dimi." or respuestaAsistenteVirtual ==
" adiós, teamy." or respuestaAsistenteVirtual == " ¡bye, timmy!" or respuestaAsistenteVirtual == " bye, timmy!"):
        vozAsistenteVirtual.hablar("Bye di0, un gusto haberte ayudado...")
        print("-----Bye di_cer0, un gusto haberte ayudado...-----")
        break

    #Si se reconoce que el usuario dice la palabra reproduce, el asistente virtual por medio de la
    #librería pywhatkit reproduce el video que el usuario dijo en YouTube.
    elif (("reproduce" or "Reproduce") in respuestaAsistenteVirtual):
        #.replace().lower(): Lo que hace el método replace() es reemplazar todas las palabras que
        #aparezcan en un string en otra cadena específica y el método lower() sirve para convertir todas
        #las letras de una cadena en minúsculas, esto es importante hacerlo porque al reproducir una
        #canción o mandar una búsqueda a internet, siempre es mejor tener puras minúsculas.
        cancionYoutube = respuestaAsistenteVirtual.replace('reproduce', '').lower()
        vozAsistenteVirtual.hablar("Ok di0, voy a reproducir una canción en YouTube")
        #pywhatkit.playonyt(): El método playonyt() de la librería pywhatkit permite reproducir un
        #video de YouTube en el navegador web predeterminado del sistema.
        pywhatkit.playonyt(cancionYoutube)
        print("-----Ok di_cer0, ya reproducí la canción del artista que pediste en YouTube...-----")

    except Exception as errorEjecucion:
        vozAsistenteVirtual.hablar("Lo siento, no escuché bien lo que dijiste debido a este error...")
        print(errorEjecucion)
        continue

#El límite de la duración de audio que puede procesar T.I.M.M.Y. en las instrucciones que se le manda es de 15
#segundos y el Mark I en sus respuestas tiene un delay de:
# - tiny: Delay de 2 segundos cuando le mando un comando hablado usando el comando.
# - base: Delay de 5 segundos cuando le mando un comando hablado usando el comando.
# - small: Delay de 10 segundos cuando le mando un comando hablado usando el comando.

```

Clase: oidoAsistente

```

#IMPORTACIÓN DE LIBRERÍAS:
import io      #io: Librería que permite la manipulación de los archivos y carpetas de nuestro ordenador.
#AudioSegment: Clase de la librería pydub que permite tomar el audio que perciba el micrófono del ordenador y
#luego que eso se pueda convertir en un archivo mp3 o wav.
import pydub   #pydub: Librería para procesar y exportar a un archivo datos de audio.
import speech_recognition as sr #sr: Librería que permite recabar audio del micrófono en tiempo real.
import whisper #whisper: Librería de OpenAI que permite hacer la transcripción de audio o video a texto.
import tempfile #tempfile: Librería que crea y maneja archivos temporales.
import os      #os: Librería que permite acceder a funciones y métodos relacionados con el sistema operativo.

```



```

#CREACIÓN DE ARCHIVO TEMPORAL:

#tempfile.mkdtemp(): El método mkdtemp() crea un directorio temporal en la carpeta predeterminada del sistema
#operativo, la cual es llamada temp y es una ubicación en el sistema operativo donde se almacenan archivos
#temporales generados por diversas aplicaciones y procesos.

archivoTemporal = tempfile.mkdtemp()

#os.path.join(): El método join() se utiliza para unir varios componentes de ruta (variables y constantes) en
#una sola ruta. Esto se utilizará para crear el archivo wav o mp3 que guardará un pedazo de la instrucción del
#usuario. Cabe mencionar que WAV es un formato de archivo de audio sin pérdidas, lo que significa que todos los
#datos de audio originales se conservan. Esto hace que los archivos WAV sean de mayor calidad que los archivos
#MP3, pero también los hace más grandes y pesados.

rutaArchivo = os.path.join(archivoTemporal, 'audioTemporal.wav')

print("Esta es la ruta del archivo temporal de las instrucciones recibidas en audio:\n" + str(rutaArchivo))


#RECONOCIMIENTO DE LAS INSTRUCCIONES DADAS AL USUARIO:

#speech_recognition.Recognizer(): El objeto Recognizer() se utiliza para crear un reconocedor de voz, el cual
#puede utilizarse para reconocer instrucciones dadas al asistente virtual por medio del habla humana.

listenerInstrucciones = sr.Recognizer()


#EscucharMicrofono: Clase propia para escuchar las instrucciones del usuario por medio del micrófono y almacenar
#eso en un archivo temporal, luego se transcribirá ese archivo de audio a texto a través del modelo Whisper y
#finalmente el texto resultante será transformado a minúsculas para que sea interpretado por el programa.

class EscucharMicrofono:

    #POO: En Python cuando al nombre de una función se le ponen dos guiones bajos antes de su nombre es porque
    #se está refiriendo a un método privado, es una buena práctica de sintaxis.

    #__escuchaInstrucciones(textoAsistente): Función propia y con modificador de acceso privado que se encarga
    #de escuchar y reconocer el audio por medio del micrófono del ordenador las palabras dichas por el usuario,
    #esto se tarda un poco en cargar la primera vez que se corre el programa debido a que el método
    #speech_recognition.Recognizer().listen() necesita un tiempo para inicializarse y configurarse; si se está
    #utilizando una computadora con poca potencia o micrófonos de baja calidad, es posible que se necesite
    #esperar más tiempo.

    def __escuchaInstrucciones(self):

        #MANEJO DE EXCEPCIONES: Es una parte de código que se conforma de dos partes, try y except:

        # - Primero se ejecuta el código que haya dentro del try y si es que llegara a ocurrir una excepción
        #   durante su ejecución, el programa brinca al código del except.

        # - En la parte de código donde se encuentra la palabra reservada except, se ejecuta cierta acción
        #   cuando ocurra el error esperado.

        #Se utiliza esta arquitectura de código cuando se quiera efectuar una acción donde se espera que pueda
        #ocurrir un error durante su ejecución.

        try:

            #with as source: La instrucción with se utiliza para definir una variable que tiene asignada un
            #método o recurso específico, el cual puede ser un archivo, una conexión a una base de datos, la
            #cámara, micrófono o cualquier otro objeto que requiera ser cerrado. La palabra clave as se utiliza
            #para asignar dicho recurso a una variable que puede utilizarse para acceder al recurso dentro del
            #bloque with.

```



```

#Dentro del manejo de excepciones la instrucción with as source: asegura que se cierre el recurso,
#incluso si se produce un error dentro del bloque try.
#speech_recognition.Microphone(): El objeto Microphone puede utilizarse para grabar audio.
with sr.Microphone() as microfono:
    print("\n\n-----Hola soy T.I.M.M.Y. Mark I tu asistente virtual, en que te puedo ayudar?-----")
    #speech_recognition.Recognizer().adjust_for_ambient_noise(): El método adjust_for_ambient_noise()
    #sirve para aplicar un filtro a la señal de sonido que quite el ruido de fondo recibido en su
    #parámetro.
    listenerInstrucciones.adjust_for_ambient_noise(microfono)
    print("Quitando ruido de fondo... ya puedes hablar.")
    #speech_recognition.Recognizer().listen(): El método listen() sirve para poder escuchar de una
    #fuente de audio en tiempo real y convertirlo a texto, para ello primero se tuvo que haber
    #instanciado la clase Recognizer.
    textoInstruccionUsuario = listenerInstrucciones.listen(microfono)
    print("Microfono encendido y escuchando...")
    #io.BytesIO(): Método que sirve para crear un flujo de bytes en memoria que puede utilizarse
    #para almacenar datos binarios, como imágenes, audio y video.
    #speech_recognition.Recognizer().listen().get_wav_data(): El método get_wav_data() se utiliza
    #para almacenar el texto recibido en el micrófono del ordenador en un flujo de datos binarios
    #que se puedan guardar en un archivo de audio con extensión wav.
    datosAudio = io.BytesIO(textoInstruccionUsuario.get_wav_data())
    #pydub.AudioSegment().from_file(): El método from_file() perteneciente a la clase AudioSegment
    #se utiliza para cargar solamente los datos de audio donde se escuche la voz del usuario,
    #logrando así que no se haga nada cuando el usuario está en silencio.
    archivoAudio = pydub.AudioSegment.from_file(datosAudio)
    print("Mandando texto del microfono a un archivo de audio wav...")
    #pydub.AudioSegment().from_file().export(path, file_type): Lo que hace el método export() es
    #exportar un segmento de audio a un archivo.
    archivoAudio.export(rutaArchivo, format = 'wav')
    print("Texto del microfono guardado en un archivo de audio temporal tipo wav...")

#Para identificar el tipo de excepción que ha ocurrido y utilizarlo en la instrucción except, se puede
#utilizar la clase Exception, que es una clase incorporada en Python utilizada para describir todos los
#tipos de excepciones, luego de colocar el nombre de la clase Exception se usa la palabra reservada "as"
#seguida de un nombre de variable, esto nos permitirá acceder a la instancia de la excepción y
#utilizarla dentro del except.
except Exception as error:
    print(error)

#Se retorna la ruta del archivo de la función porque de ahí extraerá el archivo de audio el modelo
#Whisper para procesarlo.
return rutaArchivo

#__transcripcionWhisper(): Función propia y con modificador de acceso privado que se encarga de recibir el
#archivo de audio recortado que contiene la instrucción mandada al asistente virtual para pasársela al
#modelo de Whisper para que la transcriba a texto.
def __transcripcionWhisper(self, audioRecortado):

```



#whisper.load_model(): Con el método load_model() se pueden cargar los diferentes modelos de lenguaje de la librería whisper, si se elige un modelo muy simple, se tendrá un peor desempeño, pero mientras más complejo sea el modelo, más se tardará en procesar el audio, a esto dentro de la librería se le llama #Parameter y Relative Speed, donde a un mayor número de parámetros, se tendrá mayor capacidad de #aprendizaje en el modelo y la velocidad relativa indica la rapidez con la que el modelo puede generar #texto, esta se calcula dividiendo el tiempo que tarda el modelo en generar una palabra entre el número #de palabras que genera. Las características de los modelos disponibles son las siguientes:

```
# - tiny:   Parameter = 39M,   Memoria que consume = 1GB   y Relative Speed = 32x.
# - base:   Parameter = 74M,   Memoria que consume = 1GB   y Relative Speed = 16x.
# - small:  Parameter = 244M,  Memoria que consume = 2GB   y Relative Speed = 6x.
# - medium: Parameter = 769M,  Memoria que consume = 5GB   y Relative Speed = 2x.
# - large:  Parameter = 1550M, Memoria que consume = 10GB  y Relative Speed = 1x.
```

#Cuando el programa sea ejecutado por primera vez empezará a descargar el modelo para poderlo utilizar, #este proceso tarda un poco en terminar, pero después identificará por sí solo el lenguaje y las #palabras dichas en el audio, no importando si este tenga ruido o si es de alguna canción con #instrumentos detrás.

```
Modelo = whisper.load_model("base")
```

#whisper.load_model().transcribe(): El método transcribe() sirve para escuchar el audio de un archivo y #transcribirlo a texto a través del modelo elegido con el método load_model(). El proceso de carga tarda #un poco y el tiempo que el modelo se tarda en procesar el audio depende de su duración. Los parámetros #que recibe el método son la ubicación del archivo de audio que quiere transcribir y el parámetro fp16 #que indica si el cálculo se realizará en formato de punto flotante de 16 bits (FP16), pero como en #algunas CPUs no se admite esta funcionalidad, se coloca como False, por lo que se está utilizando punto #flotante de 32 bits (FP32) en su lugar. Además cabe mencionar que el audio escuchado por el método será #recortado en cachos de 30 segundos y estos serán igualmente divididos en pedazos.

```
TranscripcionAudio_a_Texto = Modelo.transcribe(audioRecortado, language = "spanish", fp16 = False)
print("Interpretando texto guardado en un archivo temporal de audio wav con el modelo Whisper...\t")
#whisper.load_model().transcribe()["text"]: Texto traducido por Whisper de un archivo de audio.
return TranscripcionAudio_a_Texto["text"]
```

#oidoAsistenteVirtual(): Método con modificador de acceso público que permite ejecutar los métodos privados #__transcripcionWhisper() para transcribir un archivo de audio a texto y __escuchaInstrucciones() para #escuchar las instrucciones del usuario por medio del micrófono y almacenar eso en un archivo temporal. El #resultado del texto escuchado del usuario será transformado a minúsculas aplicándole el método lower(). #self: La instrucción self se utiliza para hacer referencia al objeto que se está manipulando cuando se instancia #la clase. Por eso es que a través de la misma nomenclatura de un punto se accede a los distintos atributos y/o #métodos con un objeto desde fuera de la clase.

```
def oidoAsistenteVirtual(self):
    return self.__transcripcionWhisper(self.__escuchaInstrucciones()).lower()
```

Clase: vozAsistente

```
#IMPORTACIÓN DE LIBRERÍAS:
import pyttsx3 #pyttsx3: Biblioteca que sirve para generar una voz disponible en el sistema operativo.
```

```

#VOZ UTILIZADA POR EL ASISTENTE VIRTUAL:

#pyttsx3.init(): El método pyttsx3.init() se utiliza para inicializar el motor de texto a voz, el cual dará la
#habilidad de hablar al asistente virtual, convirtiendo el texto retornado por el modelo de Whisper en audio que
#salga de la computadora.

motorVoz = pyttsx3.init()

#pyttsx3.init().getProperty("voices"): El método getProperty() se utiliza para obtener una lista de las voces
#disponibles en el motor de texto a voz incluidos en el sistema operativo del ordenador.

voz_AsistenteVirtual = motorVoz.getProperty("voices")

#Bucle for para mostrar todas las opciones de sintetizadores de voz disponibles en el sistema operativo, el
#número de opciones dependerá de los lenguajes que pueda manejar el sistema operativo de la computadora que
#digan text to speech y se pueden ver al ingresar a la opción de: Windows -> Configuración -> Hora e Idioma ->
#Idioma y Región -> Idioma -> Idiomas que diga texto a voz. Si se quiere se podría agregar más idiomas.
for i in range(len(voz_AsistenteVirtual)):
    print(voz_AsistenteVirtual[i])

#pyttsx3.init().setProperty(): Este método toma dos parámetros y no devuelve ningún valor porque indica las
#siguientes características del sintetizador de voz del asistente virtual:

# - name: El nombre de la propiedad que se desea establecer.
# - rate: Indica la velocidad de la voz en palabras por minuto.
# - volume: El volumen de la voz, de 0 a 1.
# - voices: Lista de todas las voces disponibles.
# - voice: La voz que se utilizará para hablar.
# - language: El idioma que se utilizará para hablar.
# - engine: Controlador de voz que se utilizará.
# - debug: Variable booleana que indica si se debe habilitar el modo de depuración o no.
# - value: El valor de la propiedad que se desea establecer.

motorVoz.setProperty("rate", 190)

motorVoz.setProperty("voice", voz_AsistenteVirtual[3].id)

#vozWindows: Clase propia que utiliza una de las voces incluidas en los lenguajes descargados en el sistema
#operativo Windows para lograr así que el asistente virtual hable.

class vozWindows:

    #hablar(textoAsistente): Función propia que permite al asistente virtual hablar por medio de la bocina del
    #ordenador.

    def hablar(self, textoAsistente):

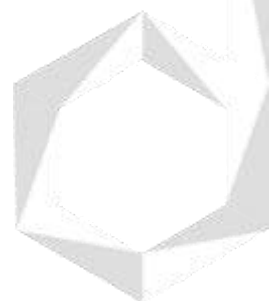
        #pyttsx3.init().say(): El método say() se utiliza para convertir texto en audio y emitirlo a través del
        #altavoz del sistema.

        motorVoz.say(textoAsistente)

        #pyttsx3.init().runAndWait(): El método runAndWait() bloquea la ejecución del programa hasta que el motor
        #de texto a voz haya terminado de hablar.

        motorVoz.runAndWait()

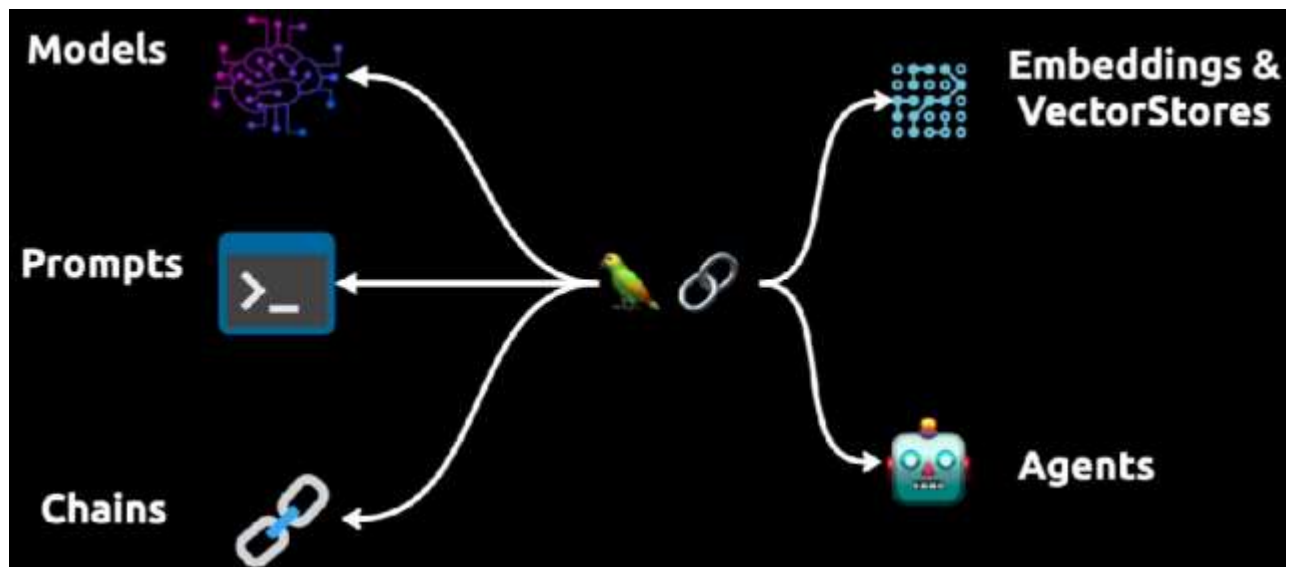
```



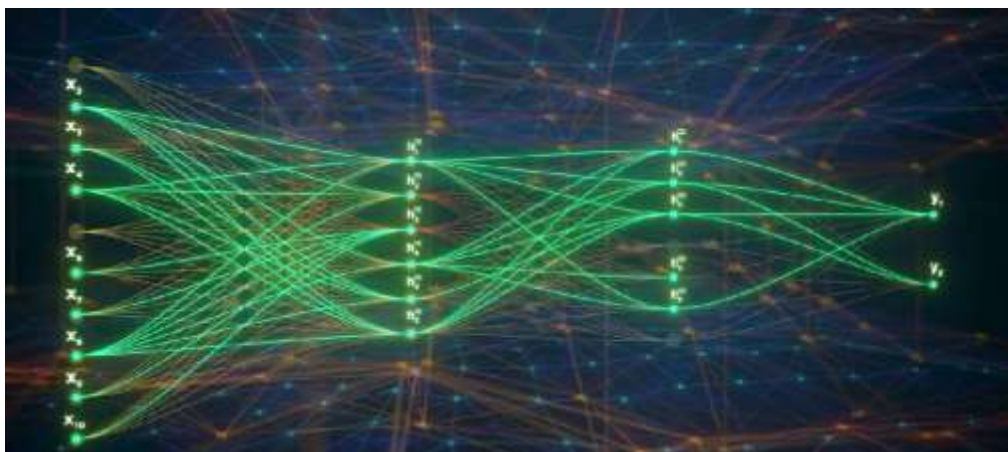
Asistente Virtual - Mark I: Hablar, YouTube, OpenAI, LangChain, Alarma, Notas y WhatsApp

LangChain 

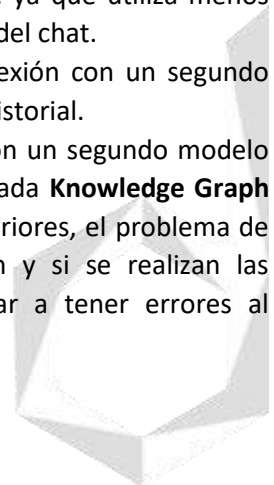
A las APIs de Google Bard o ChatGPT se les envía un texto y me retornan otro que ya haya sido procesado por el LLM, pero si se busca que estos modelos de lenguaje realicen acciones que no pueden hacer bien por sí solas como resolver ecuaciones matemáticas o que tomen información de archivos propios, bases de datos privadas, páginas web específicas o inclusive de otros modelos, ahí es donde entra en juego la herramienta open source de LangChain. Cabe mencionar que la frase “realizar acciones” no se refiere a que el modelo pueda ejecutar funciones externas al código como prender luces o cosas del estilo, sino que se refiere a que pueda contestar preguntas de mejor forma a través de distintas herramientas o fuentes de información. Para ello se hace uso de las siguientes 6 herramientas:



1. **Modelos (Models):** El modelo se refiere a la red neuronal que se va a utilizar para procesar el texto de entrada y generar una respuesta. Existen varios hasta dentro de una misma compañía, por ejemplo, OpenAI cuenta con gpt3, gpt4, gpt3.5 turbo, etc.



2. **Prompt:** Es el texto que se le envía al modelo para generar una respuesta y en este es donde se utilizan las técnicas de **Prompt Engineering** previamente explicadas, para ello la librería LangChain cuenta con diferentes clases que permiten utilizar dichas técnicas:
- **Prompt Template:** Esta clase permite crear una plantilla, utilizada cuando se quiera mandar una instrucción con algunas partes que puedan ser variables a una LLM, como por ejemplo cuando se quiere traducir cierto documento; la instrucción será la misma, pero el contenido del Prompt será distinto.
 - **FewShotTemplate:** Esta clase permite complementar la plantilla creada con ejemplos que le den contexto al LLM para que entienda el formato y tipo de respuesta que queremos que entregue, utilizando así la técnica **Few-Shot Prompting** de **Prompt Engineering**.
 - **Chat Prompt Template:** Esta clase permite armar una plantilla que contenga instrucciones constantes y contenidos variables para mandársela a un chat, que a diferencia de la clase **Prompt Template**, guardará el historial de la conversación. Para ello hace uso de 3 clases con el fin de referirse al rol que adopta el chat al responder, el mensaje mandado por el usuario y la respuesta que devuelve, de la misma forma como se hace con la API de ChatGPT.
 - **System Message Prompt Template:** Con esta clase se representa el rol que indica a ChatGPT a quién está interpretando cuando responda las preguntas del usuario.
 - **Human Message Prompt Template:** Con esta clase se representa el rol del usuario que manda preguntas a ChatGPT.
 - **AI Message Prompt Template:** Con esta clase se representa el rol que es adoptado por ChatGPT siempre que responda la pregunta de un usuario. Su mayor uso es el de permitir que el chat recuerde entradas y salidas anteriores.
 - **Output Parser:** Esta clase permite dar cierto formato a la respuesta recibida de un LLM, ya sea para ordenarla en forma de lista, diccionario, XML, JSON, etc.
3. **Memoria (Memory):** Permite almacenar las preguntas y respuestas hechas entre el LLM y el usuario, permitiendo así que se simule una conversación entre ambos, para ello la librería LangChain cuenta con diferentes clases que permiten almacenar la información del chat de distintas formas:
- **ConversationBufferMemory:** Clase que guarda todo el historial, o sea cada cosa que haya dicho el o los usuarios del chat y todo lo que haya contestado el modelo.
 - **ConversationBufferWindowMemory:** Clase que guarda solo los últimos mensajes del historial, esto se hace para que el costo del uso de la API baje, ya que utiliza menos tokens, pero se corre el riesgo de perder información importante del chat.
 - **ConversationSummaryMemory:** Clase que a través de una conexión con un segundo modelo (Cadena) crea y guarda un resumen en inglés de todo el historial.
 - **ConversationKGMemory:** Clase que a través de una conexión con un segundo modelo (Cadena) crea y guarda una lista de palabras clave del chat llamada **Knowledge Graph** para darle contexto al modelo cuando responda preguntas posteriores, el problema de usar esta herramienta es que tiene problemas de traducción y si se realizan las preguntas en un lenguaje que no sea inglés, se puede llegar a tener errores al interpretar el historial.

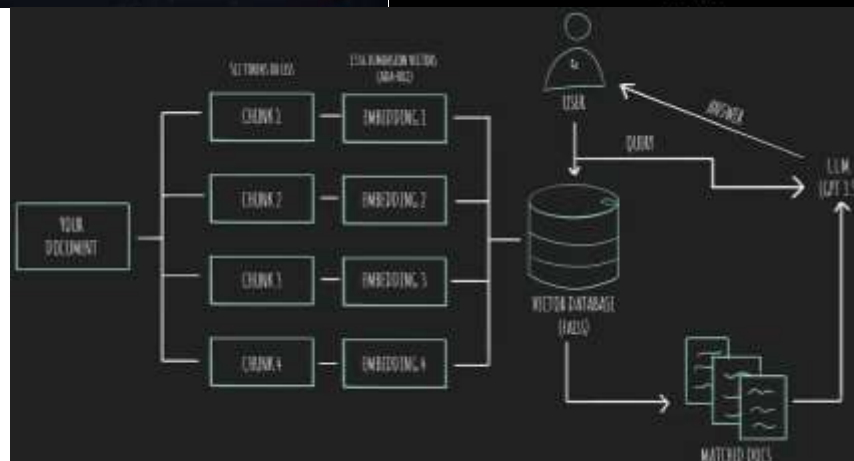
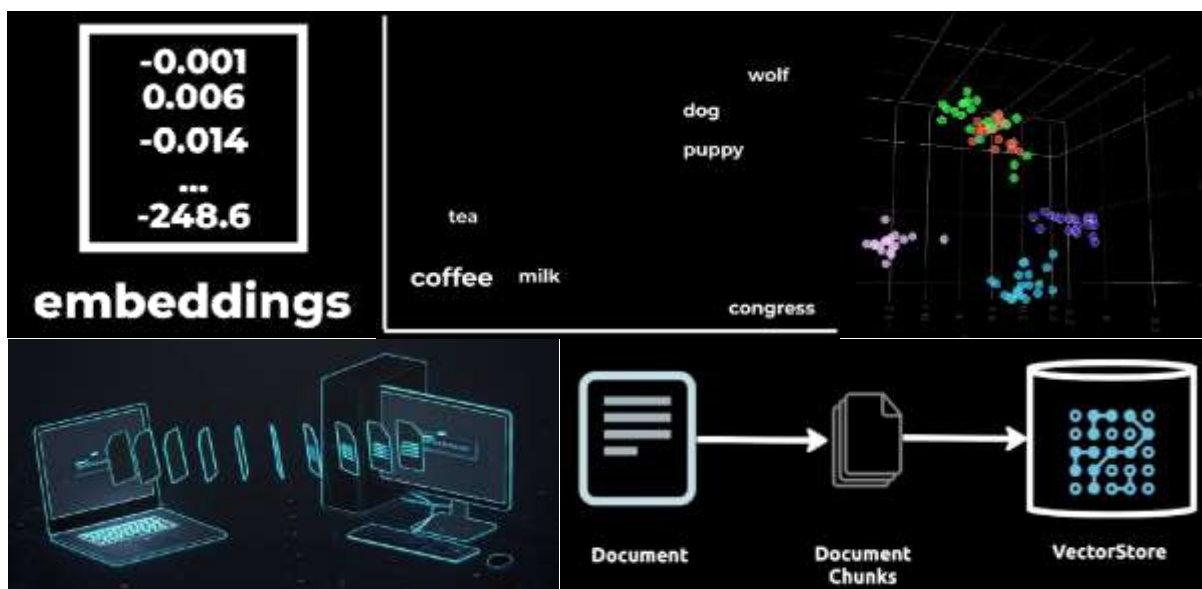


4. **Cadenas (Chains):** Este es de los conceptos más importantes de la librería, ya que permite conectar varios modelos entre sí, hasta cuando son de distintos tipos, permitiéndonos así realizar varias iteraciones entre modelos durante una consulta para obtener un mejor procesamiento final de los datos cuando este se busca aplicar a tareas muy complejas, existen dos tipos globales de cadenas:
- **LLMChain:** Con esta clase se conecta un prompt con un modelo de lenguaje, creando así una cadena individual.
 - **SequentialChain:** Con esta clase se pueden conectar dos o más cadenas individuales (osea modelos de lenguaje que se encuentran enlazados con un prompt), pudiendo recibir así múltiples entradas y generar múltiples salidas, ya que la salida de una cadena puede ser la entrada de otra cadena de forma secuencial.
 - **SimpleSequentialChain:** Con esta clase que se realiza lo mismo que con la cadena **SequentialChain** pero con la condición de que solo puede recibir 1 entrada y proporcionar 1 salida.

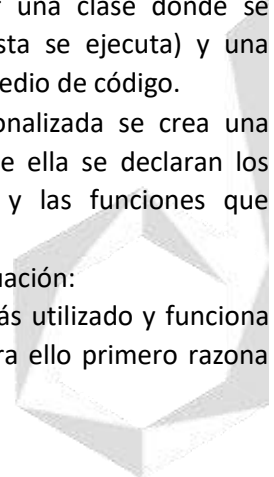


5. **Índices (Retrieval o Data connection):** La forma en la que más se aprovechan los modelos de lenguaje es cuando se les da acceso a distintas fuentes de información, como lo puede ser un archivo PDF, Word, Excel, PowerPoint, etc. Los índices en LangChain son los que nos van a permitir enlazar un gran número de documentos para que sean procesados por el modelo, para ello la librería cuenta con diferentes clases que permiten realizar el enlace:
- **DirectoryLoader:** Esta clase permite abrir, cargar y procesar todos los archivos de un mismo tipo que se encuentren en una carpeta para después pasárselos a los LLM y que sean procesados, ya sea que tengan extensión PDF, txt, Word, etc.
 - **CharacterTextSplitter:** Como los modelos solo aceptan un número finito de Tokens (pedazos de palabras), lo que se hace para que el LLM pueda procesar toda esa información es dividirla en cachos llamados Chunks, lo cual es realizado por esta clase.
 - **PyPDFLoader:** Clase de la librería langchain que permite ingresar un archivo pdf al programa y dividirlo en función de su número de páginas.
 - **PdfReader:** Clase de la librería PyPDF2 que permite ingresar un archivo pdf al programa y dividirlo en función de su número de páginas y luego el contenido de sus páginas lo divide en cachos a través de la función **CharacterTextSplitter**.
 - **OpenAIEmbeddings:** Los LLM asocian las palabras que reciben a través de un vector llamado **Embedding**, el cual es un simple array de varias dimensiones que se encuentra en un espacio vectorial, cuya función es asociar de forma gráfica una palabra con otras parecidas y/o alejarla de otras que sean muy distintas, de esta manera es como el modelo entiende el lenguaje humano. Si este proceso de conversión de un chunk de palabras a un embedding se realiza a través de OpenAI, es cobrado por medio de la API Key en función del número de Tokens que son convertidos a vectores numéricos.

- **FAISS y Chroma:** Las clases FAISS y Chroma representan dos tipos de **Vector Stores** de OpenAI, las cuales son bases de datos optimizadas para almacenar los vectores obtenidos después de procesar Chunks de información.
 - **FAISS:** Vector store de rápida reacción, poco flexible y difícil de usar.
 - **Chroma:** Vector store de lenta reacción, flexible y fácil de usar.
- **Retriever:** Una vez que los datos de una fuente externa a OpenAI hayan sido convertidos a vectores y luego almacenados en una base de datos, se podrá realizar consultas al modelo y este contestará en función de la información contenida en los documentos anexados.
 - **RetrievalQA:** Esta clase permite traer información de alguna fuente en específico para contestar una pregunta hecha a un modelo LLM, sabiendo a dónde tiene que ir a buscar para obtener la respuesta solicitada.
 - **ConversationalRetrievalChain:** Esta clase permite traer información de alguna fuente en específico para contestar una pregunta hecha a un modelo de Chat, sabiendo a dónde tiene que ir a buscar para obtener la respuesta solicitada, además de traer de vuelta de qué documento y chunk obtuvo la información requerida para contestar la pregunta hecha por el usuario.



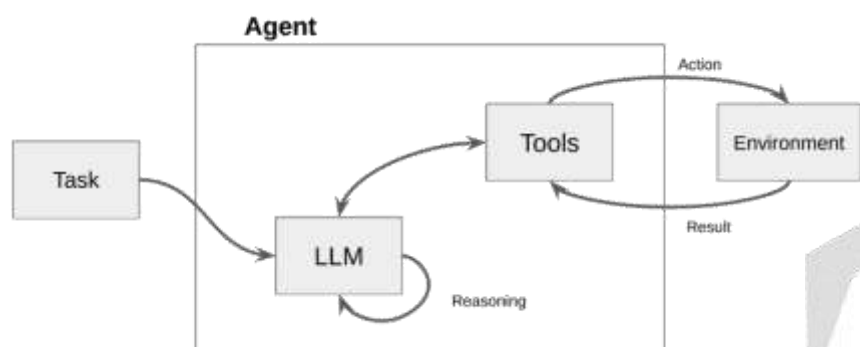
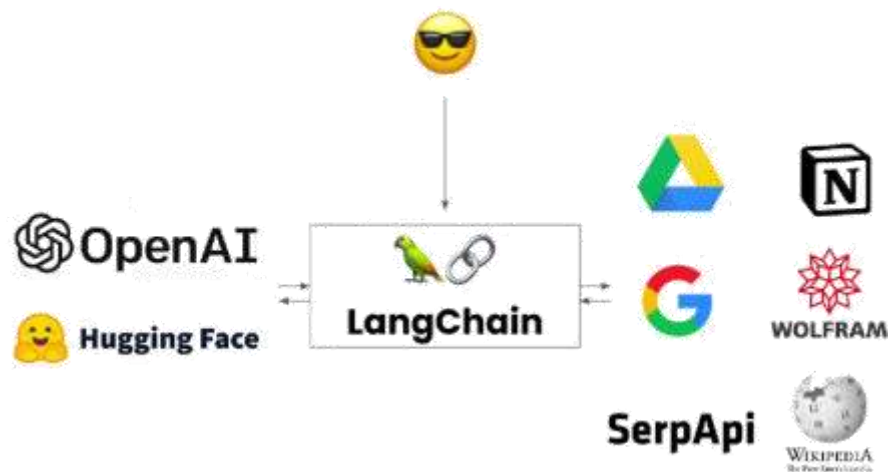
6. **Agentes (Agents):** Estos son modelos o cadenas a las cuales se les da acceso a una fuente o API para que puedan realizar alguna acción que no pueda ser bien ejecutada por el LLM, solucionando así una tarea específica, como obtener datos meteorológicos, resolver ecuaciones, matemáticas, etc. Esto se realiza a través de las siguientes herramientas externas:
- **Tools:** Son las herramientas con las que ya cuenta langchain para realizar una acción.
 - **Ejecutar comandos en consola:**
 - **terminal:** Herramienta que permite ejecutar comandos en consola.
 - **python_repl:** Esta herramienta permite ejecutar solamente scripts (programas) de Python a través de la consola del sistema.
 - **Realizar búsquedas en internet:**
 - **serpapi:** Tool que permite extraer información de internet para responder una realizada por el usuario.
 - **google-search:** Esta herramienta de langchain permite utilizar específicamente el buscador de Google para obtener la información que responde una pregunta realizada al agente.
 - **wikipedia:** Tool que permite buscar información en Wikipedia.
 - **requests:** Esta herramienta permite extraer información de la URL de un sitio web en específico para responder una pregunta.
 - **Resolver o contestar preguntas acerca de operaciones matemáticas:** Cuando se usen estas tools es recomendable declarar que la temperatura del modelo sea de 0, para que siempre dé el mismo resultado.
 - **wolfram-alpha:** Esta herramienta permite resolver problemas o contestar preguntas que tengan que ver con matemáticas, ciencia, tecnología, etc.
 - **pal-math:** Esta herramienta permite resolver problemas matemáticos a través de una instrucción, como crear ecuaciones a través de un problema de la vida real y cosas por el estilo.
 - **llm-math:** Esta herramienta permite resolver problemas matemáticos.
 - **Obtener información meteorológica:**
 - **open-meteo-api:** Permite obtener información meteorológica a través de la herramienta OpenMeteo.
 - **Obtener información de noticias recientes o películas:**
 - **news-api:** Obtiene información acerca de noticias actuales.
 - **tmdb-api:** Obtiene información acerca de películas.
 - **Herramientas personalizadas:** Para ello se debe crear una clase donde se declare un nombre, descripción (que indica cuando esta se ejecuta) y una función propia que describe que la acción a realizar por medio de código.
 - **BaseTool:** Para declarar una herramienta personalizada se crea una clase propia que herede de BaseTool, dentro de ella se declaran los valores de los parámetros name, description y las funciones que ejecutan la acción personalizada.
 - **Agente:** Existen los diferentes tipos de agentes descritos a continuación:
 - **zero-shot-react-description:** Este tipo de agente es el más utilizado y funciona con un modelo LLM, por lo que no tendrá memoria. Para ello primero razona



sobre la pregunta que se le hizo, luego recopila información de las herramientas que tenga disponibles y finalmente contesta algo.

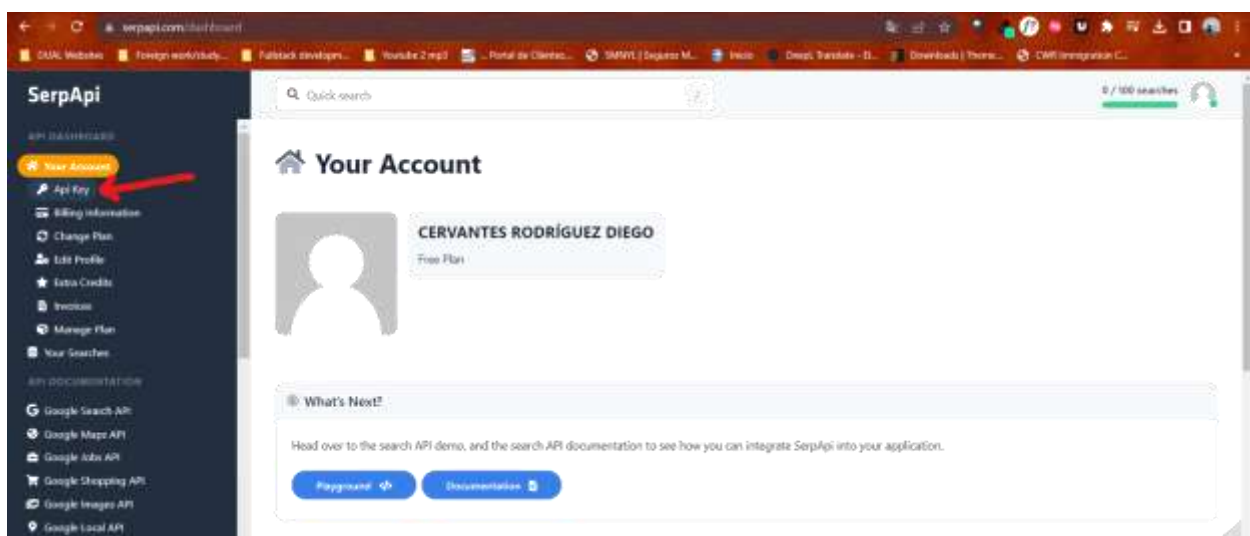
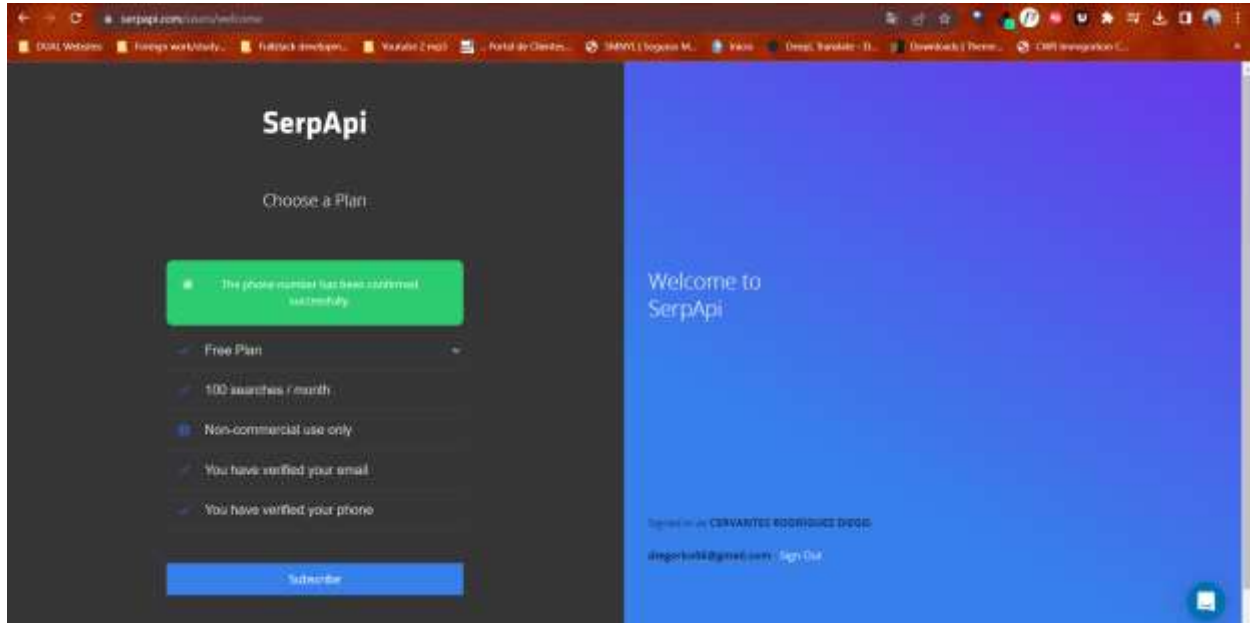
- **conversational-react-description:** Este tipo de agente funciona con un modelo de Chat, por lo que en este caso sí se guardará el historial de la conversación a través de una variable de memoria. Para ello primero razona sobre la pregunta que se le hizo, luego recopila información de las herramientas que tenga disponibles y finalmente contesta algo.
- **react-docstore:** Este tipo de agente está hecho para interactuar con mucha información extraída de documentos o artículos extraídos de buscadores como Wikipedia o Google que ya deben estar anexados al modelo por medio de índices, para que a través de ellos conteste las preguntas hechas por el usuario.
- **self-ask-with-search:** Este tipo de agente lo que hace es realizarse preguntas intermedias a sí mismo que tengan que ver con la pregunta hecha por el usuario, luego investiga la respuesta de dichas preguntas de forma individual en un buscador y utiliza las respuestas encontradas para responder la pregunta principal. Debido a su funcionamiento, forzosamente debe tener integrada una herramienta que le permita realizar búsquedas en internet.

El mejor tipo de agente a elegir dependerá de las necesidades específicas del proyecto. Los agentes **zero-shot-react-description** y **conversational-react-description** son buenos al realizar tareas generales, el agente **react-docstore** es mejor utilizarlo para interactuar con un almacén de documentos y el agente **self-ask-with-search** es una buena opción para realizar búsquedas en la web.



Para poder utilizar la Tool **serpapi** que permite realizar búsquedas en Google se debe activar la SerpApi, para ello ingresamos en el siguiente enlace, creamos nuestra cuenta y damos clic en la opción de Api Key para que esta la copiemos y peguemos en el código:

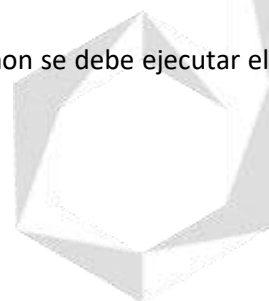
<https://serpapi.com/users/welcome>



Instalaciones:

- **LangChain:** Para poder utilizar la librería LangChain en programas de Python se debe ejecutar el siguiente comando en la consola CMD de Windows.

```
pip install langchain
```



```
C:\WINDOWS\system32\cmd. x + -
C:\Users\diego>pip install langchain
Collecting langchain
  Obtaining dependency information for langchain from https://files.pythonhosted.org/packages/64/f8/873d88338fb8aa479664d8296d73ee4652284aa58ff388648b2f2978f8d3/langchain-0.0.268-py3-none-any.whl.metadata
  Downloading langchain-0.0.268-py3-none-any.whl.metadata (14 kB)
Collecting PyYAML>=5.1 (from langchain)
  Obtaining dependency information for PyYAML>=5.1 from https://files.pythonhosted.org/packages/84/4d/8270d1ab9290b03da94e5425f5e87306b990f47e8e88f3a92c158402bf/PyYAML-6.0.1-cp39-cp39-win_amd64.whl.metadata
  Downloading PyYAML-6.0.1-cp39-cp39-win_amd64.whl.metadata (2.1 kB)
Collecting SQLAlchemy>=1.4 (from langchain)
  Obtaining dependency information for SQLAlchemy>=1.4 from https://files.pythonhosted.org/packages/01/fb/becd158acfec58b7a34d8d4ff979df9f8e5f95aa2114b5dacf58a8752f20/SQLAlchemy-2.0.20-cp39-cp39-win_amd64.whl.metadata
  Downloading SQLAlchemy-2.0.20-cp39-cp39-win_amd64.whl.metadata (9.7 kB)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from langchain) (3.8.5)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from langchain) (4.0.1)
Collecting dataclasses-json<0.6.0,>=0.5.7 (from langchain)
  Obtaining dependency information for dataclasses-json<0.6.0,>=0.5.7 from https://files.pythonhosted.org/packages/97/5f/e7cc9f3615283f8cab8b6c9c1125e8bcb9d76f8b301801b5b877b286c/dataclasses-json-0.5.14-py3-none-any.whl.metadata
  Downloading dataclasses-json-0.5.14-py3-none-any.whl.metadata (22 kB)
Collecting langsmith<0.1.0,>=0.0.21 (from langchain)
  Obtaining dependency information for langsmith<0.1.0,>=0.0.21 from https://files.pythonhosted.org/packages/d7/47/0b36622cbb352fdd75f3216c3a8ea254f38ee1e2abfd56c215634829d/langsmith-0.0.25-py3-none-any.whl.metadata
  Downloading langsmith-0.0.25-py3-none-any.whl.metadata (10 kB)
Collecting numexpr<3.0.0,>=2.8.4 (from langchain)
  Obtaining dependency information for numexpr<3.0.0,>=2.8.4 from https://files.pythonhosted.org/packages/41/17/22c118d3935e7301b0b33e9b6f5833cf2d86773d6cd89fdb1bb1214018/numexpr-2.8.5-cp39-cp39-win_amd64.whl.metadata
  Downloading numexpr-2.8.5-cp39-cp39-win_amd64.whl.metadata (8.2 kB)
Requirement already satisfied: numpy<2,>=1 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from langchain) (1.21.2)
Collecting pydantic<3,>=1 (from langchain)
  Obtaining dependency information for pydantic<3,>=1 from https://files.pythonhosted.org/packages/fd/35/86b1e7571e695587df86df2937180436dc0c4a377d2f816d4e47673791a/pydantic-2.1.1-py3-none-any.whl.metadata
  Downloading pydantic-2.1.1-py3-none-any.whl.metadata (145 kB)
145.6/145.6 kB 221.3 MB/s eta 0:00:00
Requirement already satisfied: requests<3,>=2 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from langchain) (2.31.0)
Collecting tenacity<9.0.0,>=8.1.0 (from langchain)
  Obtaining dependency information for tenacity<9.0.0,>=8.1.0 from https://files.pythonhosted.org/packages/f4/f1/990741d5bb2487d529d20a433218ffa136a367751e456214813b481c4075/tenacity-8.2.3-py3-none-any.whl.metadata
  Downloading tenacity-8.2.3-py3-none-any.whl.metadata (1.0 kB)
Requirement already satisfied: attrs>=17.1.0 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (23.1.0)
```

- **Prompt - TikTok:** Si se quiere calcular el número de Tokens mandados en un Prompt de LangChain se debe ejecutar el siguiente comando en la consola CMD de Windows para realizar la instalación de la librería tiktoken.

```
pip install tiktoken
```

```
C:\WINDOWS\system32\cmd. x + -
C:\Users\diego>pip install tiktoken
Collecting tiktoken
  Downloading tiktoken-0.4.0-cp39-cp39-win_amd64.whl (635 kB)
635.6/635.6 kB 1.0 MB/s eta 0:00:00
Collecting regex>=2022.1.18 (from tiktoken)
  Obtaining dependency information for regex>=2022.1.18 from https://files.pythonhosted.org/packages/04/30/12624697d49c42a4f011935c9948c3d829eadd01e20966ec5ae7556d84c/regex-2023.8.8-cp39-cp39-win_amd64.whl.metadata
  Downloading regex-2023.8.8-cp39-cp39-win_amd64.whl.metadata (42 kB)
42.0/42.0 kB ? eta 0:00:00
Requirement already satisfied: requests>=2.26.0 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from tiktoken) (2.31.0)
```

- **Memoria - Knowledge Graph:** Si se quiere conectar distintos modelos para guardar en memoria las palabras clave del historial de una conversación con la clase **ConversationKGMemory** de la librería LangChain se debe ejecutar el siguiente comando en la consola CMD de Windows para realizar la instalación de la librería networkx.


```
pip install networkx
```

```
C:\WINDOWS\system32\cmd. x + -
C:\Users\diego>pip install networkx
Collecting networkx
  Downloading networkx-3.1-py3-none-any.whl (2.1 MB)
2.1/2.1 MB 3.0 MB/s eta 0:00:00
Installing collected packages: networkx
Successfully installed networkx-3.1
C:\Users\diego>
```



- **Índices - PdfReader - PyPDF2:** Si se quiere leer el contenido de un pdf con una herramienta fuera de la librería LangChain se debe ejecutar el siguiente comando en la consola CMD de Windows para realizar la instalación de la librería PyPDF2.

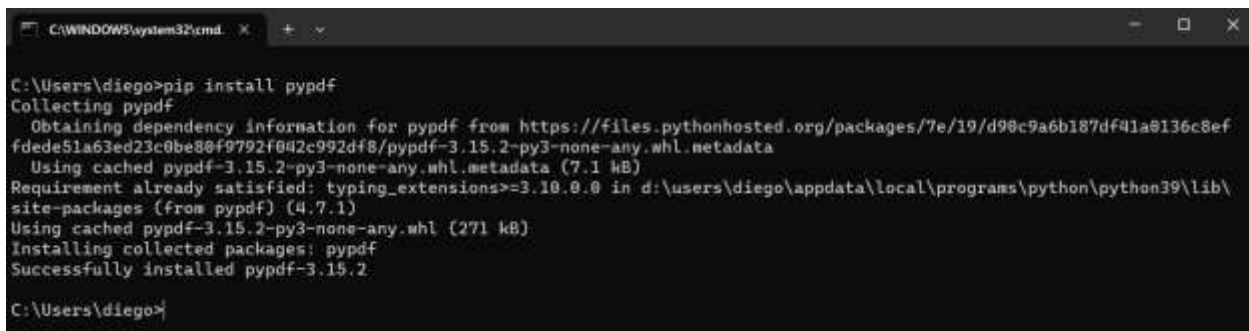
```
pip install PyPDF2
```



```
C:\WINDOWS\system32\cmd. X + -
C:\Users\diego>pip install PyPDF2
Collecting PyPDF2
  Using cached pypdf2-3.0.1-py3-none-any.whl (232 kB)
Requirement already satisfied: typing_extensions>=3.10.0.0 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from PyPDF2) (4.7.1)
Installing collected packages: PyPDF2
Successfully installed PyPDF2-3.0.1
C:\Users\diego>
```

- **Índices - PyPDFLoader - LangChain:** Si se quiere leer el contenido de un pdf a través de una herramienta de índices perteneciente a la biblioteca LangChain se debe ejecutar el siguiente comando en la consola CMD de Windows para realizar la instalación de la librería pypdf.

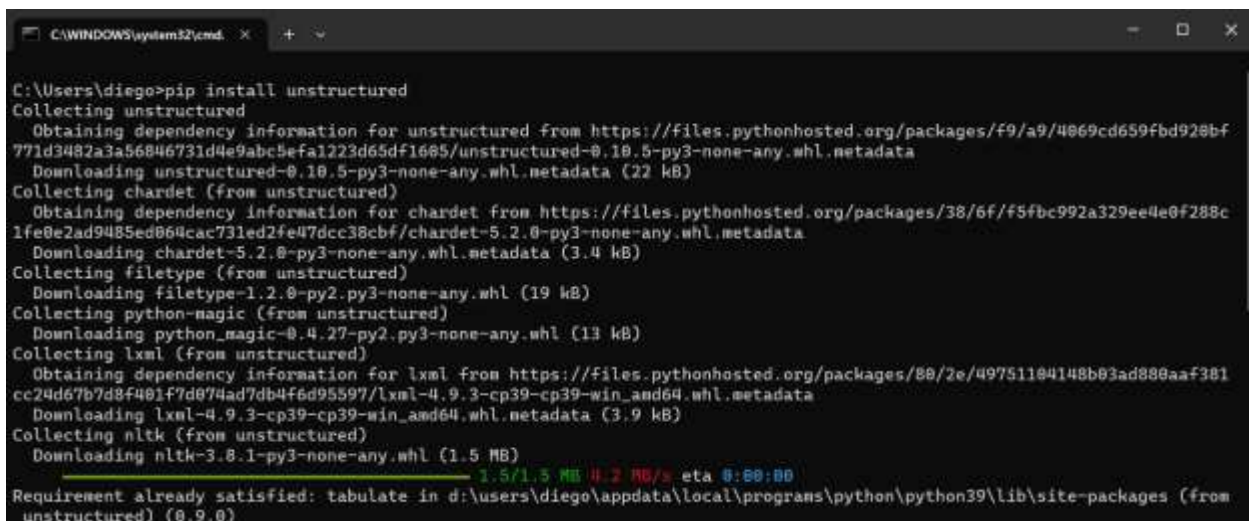
```
pip install pypdf
```



```
C:\WINDOWS\system32\cmd. X + -
C:\Users\diego>pip install pypdf
Collecting pypdf
  Obtaining dependency information for pypdf from https://files.pythonhosted.org/packages/7e/19/d90c9a6b187df41a0136c8ef-fde51a63ed23c0be80f9792f042c992df8/pypdf-3.15.2-py3-none-any.whl.metadata
  Using cached pypdf-3.15.2-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: typing_extensions>=3.10.0.0 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pypdf) (4.7.1)
Using cached pypdf-3.15.2-py3-none-any.whl (271 kB)
Installing collected packages: pypdf
Successfully installed pypdf-3.15.2
C:\Users\diego>
```

- **Índices - DirectoryLoader (.txt) - LangChain:** Para poder leer todos los documentos con extensión txt de un directorio en específico se debe ejecutar el siguiente comando en la consola CMD de Windows para realizar la instalación de la librería unstructured.

```
pip install unstructured
```

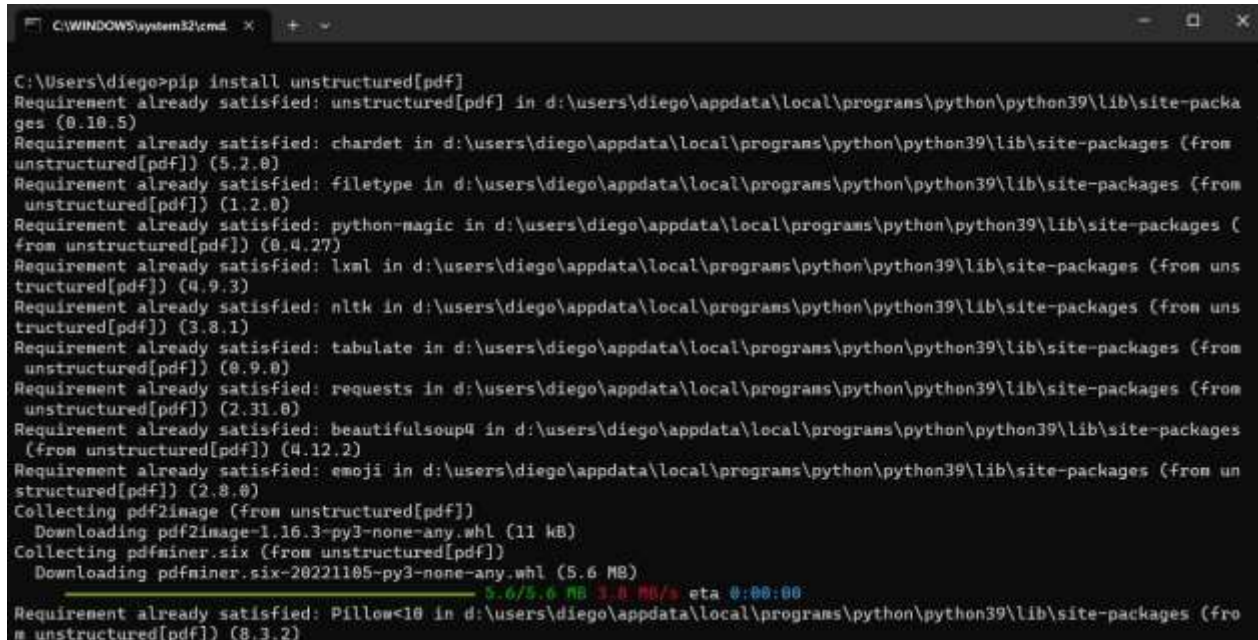


```
C:\WINDOWS\system32\cmd. X + -
C:\Users\diego>pip install unstructured
Collecting unstructured
  Obtaining dependency information for unstructured from https://files.pythonhosted.org/packages/f9/a9/4069cd659fbd920bf-771d3482a3a56846731d4e9abc5efal223d65df1605/unstructured-0.10.5-py3-none-any.whl.metadata
  Downloading unstructured-0.10.5-py3-none-any.whl.metadata (22 kB)
Collecting chardet (from unstructured)
  Obtaining dependency information for chardet from https://files.pythonhosted.org/packages/38/6f/f5fbc992a329ee4e0f288c1fe0e2ad9485ed064cac731ed2fe47dccc38cbf/chardet-5.2.0-py3-none-any.whl.metadata
  Downloading chardet-5.2.0-py3-none-any.whl.metadata (3.4 kB)
Collecting filetype (from unstructured)
  Downloading filetype-1.2.0-py2.py3-none-any.whl (19 kB)
Collecting python-magic (from unstructured)
  Downloading python_magic-0.4.27-py2.py3-none-any.whl (13 kB)
Collecting lxml (from unstructured)
  Obtaining dependency information for lxml from https://files.pythonhosted.org/packages/00/2e/49751104148b03ad808aaf301cc24d67b7d8f401f7d074ad7db4f6d95597/lxml-4.9.3-cp39-cp39-win_amd64.whl.metadata
  Downloading lxml-4.9.3-cp39-cp39-win_amd64.whl.metadata (3.9 kB)
Collecting nltk (from unstructured)
  Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
Requirement already satisfied: tabulate in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured) (0.9.0)
1.5/1.5 MB 4.2 MB/s eta 0:00:00
```



- **Índices - DirectoryLoader (.pdf) - LangChain:** Para poder leer todos los documentos con extensión pdf de un directorio en específico se debe ejecutar el siguiente comando en la consola CMD de Windows para realizar la instalación de la librería unstructured [pdf].

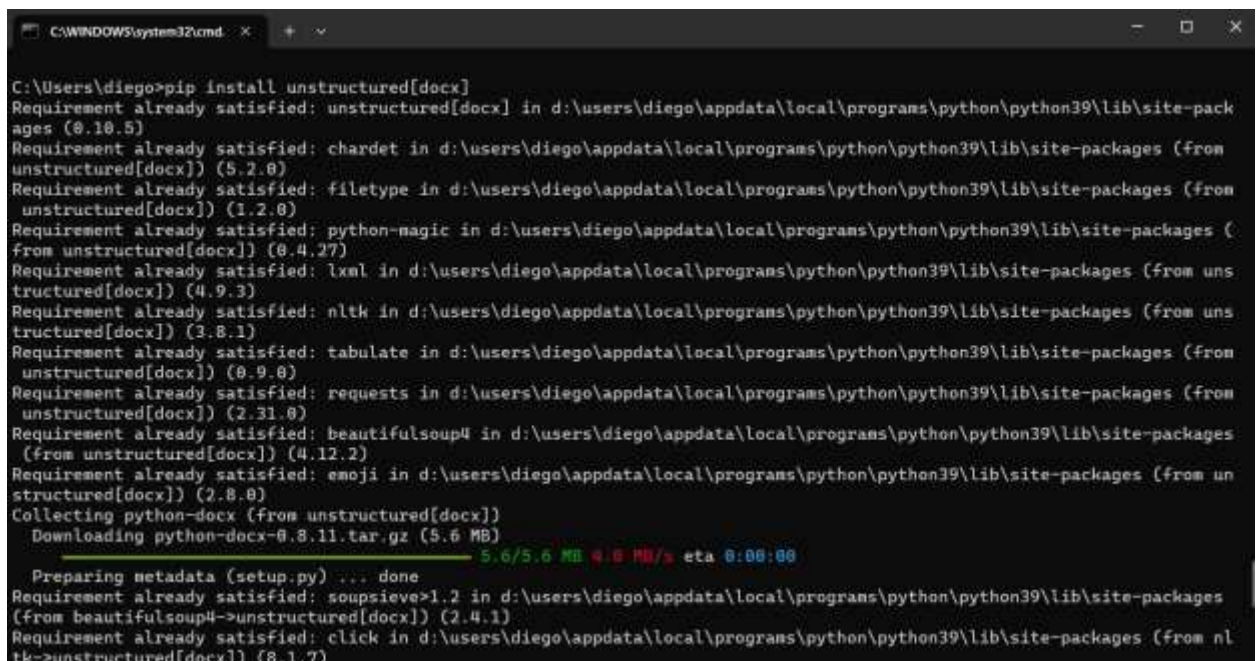
```
pip install unstructured[pdf]
```



```
C:\Users\diego>pip install unstructured[pdf]
Requirement already satisfied: unstructured[pdf] in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (0.10.5)
Requirement already satisfied: chardet in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[pdf]) (5.2.0)
Requirement already satisfied: filetype in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[pdf]) (1.2.0)
Requirement already satisfied: python-magic in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[pdf]) (0.4.27)
Requirement already satisfied: lxml in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[pdf]) (4.9.3)
Requirement already satisfied: nltk in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[pdf]) (3.8.1)
Requirement already satisfied: tabulate in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[pdf]) (0.9.0)
Requirement already satisfied: requests in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[pdf]) (2.31.0)
Requirement already satisfied: BeautifulSoup4 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[pdf]) (4.12.2)
Requirement already satisfied: emoji in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[pdf]) (2.8.0)
Collecting pdf2image (from unstructured[pdf])
  Downloading pdf2image-1.16.3-py3-none-any.whl (11 kB)
Collecting pdfminer.six (from unstructured[pdf])
  Downloading pdfminer.six-20221105-py3-none-any.whl (5.6 MB)
    5.6/5.6 MB 3.0 MB/s eta 0:00:00
Requirement already satisfied: Pillow<10 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[pdf]) (8.3.2)
```

- **Índices - DirectoryLoader (.docx) - LangChain:** Para poder leer todos los documentos con extensión docx (Word) de un directorio en específico se debe ejecutar el siguiente comando en la consola CMD de Windows para realizar la instalación de la librería unstructured [docx].

```
pip install unstructured[docx]
```



```
C:\Users\diego>pip install unstructured[docx]
Requirement already satisfied: unstructured[docx] in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (0.10.5)
Requirement already satisfied: chardet in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[docx]) (5.2.0)
Requirement already satisfied: filetype in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[docx]) (1.2.0)
Requirement already satisfied: python-magic in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[docx]) (0.4.27)
Requirement already satisfied: lxml in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[docx]) (4.9.3)
Requirement already satisfied: nltk in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[docx]) (3.8.1)
Requirement already satisfied: tabulate in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[docx]) (0.9.0)
Requirement already satisfied: requests in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[docx]) (2.31.0)
Requirement already satisfied: BeautifulSoup4 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[docx]) (4.12.2)
Requirement already satisfied: emoji in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from unstructured[docx]) (2.8.0)
Collecting python-docx (from unstructured[docx])
  Downloading python-docx-0.8.11.tar.gz (5.6 MB)
    5.6/5.6 MB 4.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: soupsieve>1.2 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from BeautifulSoup4->unstructured[docx]) (2.4.1)
Requirement already satisfied: click in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from nltk->unstructured[docx]) (8.1.7)
```



- **Índices - FAISS - Vector Stores:** Para almacenar los embeddings (vectores numéricos) que representan los datos pertenecientes a los archivos anexados al programa en un **Vector Store** de tipo **FAISS** se debe ejecutar el siguiente comando en la consola CMD de Windows.

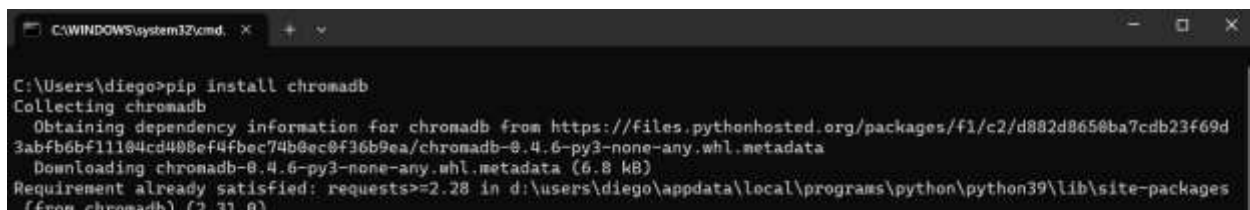
```
pip install faiss-cpu
```



```
C:\Users\diego>pip install faiss-cpu
Collecting faiss-cpu
  Downloading faiss_cpu-1.7.4-cp39-cp39-win_amd64.whl (10.8 MB)
    10.8/10.8 MB 3.1 MB/s eta 0:00:00
Installing collected packages: faiss-cpu
Successfully installed faiss-cpu-1.7.4
C:\Users\diego>
```

- **Índices - Chroma - Vector Stores:** Para almacenar los embeddings (vectores numéricos) que representan los datos pertenecientes a los archivos anexados al programa en un **Vector Store** de tipo **Chroma** se debe ejecutar el siguiente comando en la consola CMD de Windows.

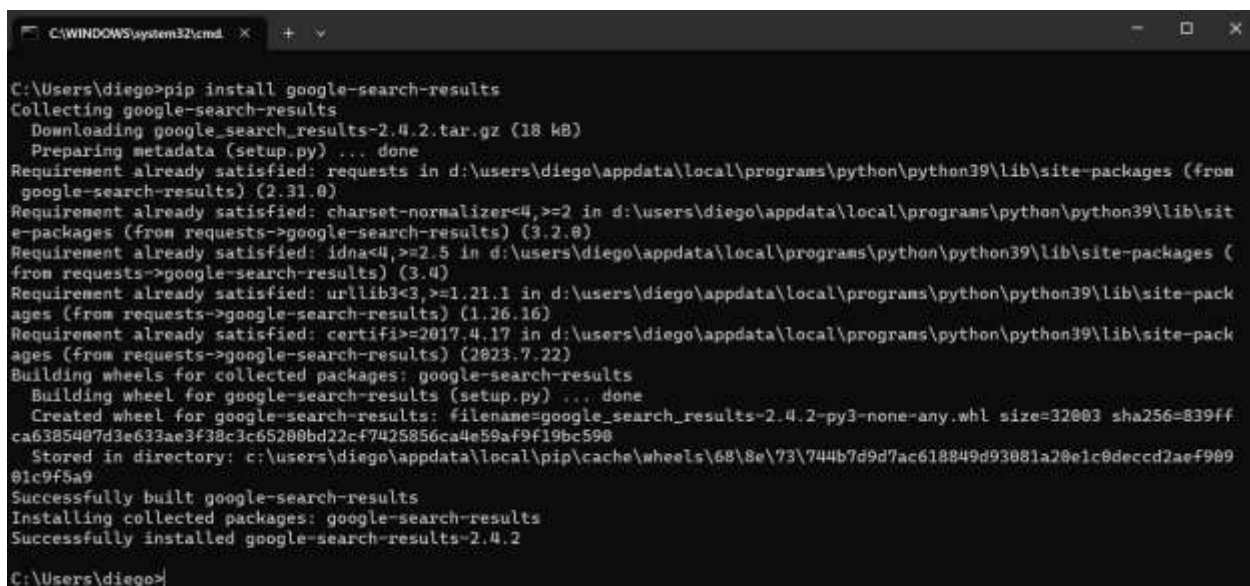
```
pip install chromadb
```



```
C:\Users\diego>pip install chromadb
Collecting chromadb
  Obtaining dependency information for chromadb from https://files.pythonhosted.org/packages/f1/c2/d882d8650ba7cdb23f69d3abfb6bf1104cd408ef4fbec74b0ec0f36b9ea/chromadb-0.4.6-py3-none-any.whl.metadata
  Downloading chromadb-0.4.6-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: requests>=2.28 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from chromadb) (2.31.0)
```

- **Agentes - SerpApi - Google Search Results:** Si se quiere proporcionar a un agente la habilidad de realizar búsquedas en internet a través de la herramienta SERPAPI se debe ejecutar el siguiente comando en la consola CMD de Windows para realizar la instalación del buscador a través de la librería Google Search Results.

```
pip install google-search-results
```

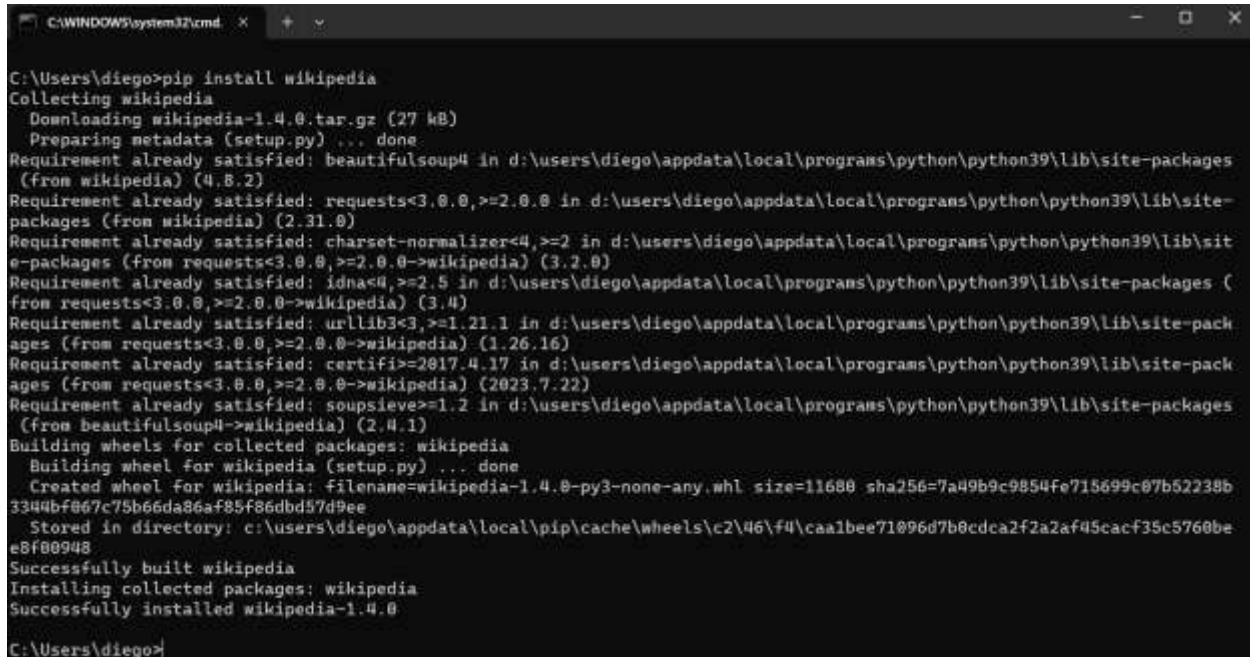


```
C:\Users\diego>pip install google-search-results
Collecting google-search-results
  Downloading google_search_results-2.4.2.tar.gz (18 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: requests in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from google-search-results) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests->google-search-results) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests->google-search-results) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests->google-search-results) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests->google-search-results) (2023.7.22)
Building wheels for collected packages: google-search-results
  Building wheel for google-search-results (setup.py) ... done
  Created wheel for google-search-results: filename=google_search_results-2.4.2-py3-none-any.whl size=32003 sha256=839ffc6385487d3e633ae3f38c3c65280bd22cf7425856ca4e59af9f19bc590
  Stored in directory: c:\users\diego\appdata\local\pip\cache\wheels\68\8e\73\744b7d9d7ac618849d93081a28e1c0decc2ae9f90901c9f5a9
Successfully built google-search-results
Installing collected packages: google-search-results
Successfully installed google-search-results-2.4.2
C:\Users\diego>
```



- **Agentes - SerpApi - Wikipedia:** Si se quiere proporcionar a un agente la habilidad de realizar búsquedas en el sitio de Wikipedia a través de la herramienta SERPAPI se debe ejecutar el siguiente comando en la consola CMD de Windows.

```
pip install wikipedia
```



```
C:\WINDOWS\system32\cmd. X + v
C:\Users\diego>pip install wikipedia
Collecting wikipedia
  Downloading wikipedia-1.4.0.tar.gz (27 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: BeautifulSoup4 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from wikipedia) (4.8.2)
Requirement already satisfied: requests<3.0.0,>=2.0.0 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from wikipedia) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests<3.0.0,>=2.0.0->wikipedia) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests<3.0.0,>=2.0.0->wikipedia) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests<3.0.0,>=2.0.0->wikipedia) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests<3.0.0,>=2.0.0->wikipedia) (2023.7.22)
Requirement already satisfied: soupsieve>=1.2 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from BeautifulSoup4->wikipedia) (2.4.1)
Building wheels for collected packages: wikipedia
  Building wheel for wikipedia (setup.py) ... done
  Created wheel for wikipedia: filename=wikipedia-1.4.0-py3-none-any.whl size=11680 sha256=7a49b9c9854fe715699c07b52238b3344bf067c75b66da86af85f86dbd57d9ee
  Stored in directory: c:\users\diego\appdata\local\pip\cache\wheels\c2\46\f4\ca1bee71096d7b0cdca2f2a2af45cacf35c5760be8f00948
Successfully built wikipedia
Installing collected packages: wikipedia
Successfully installed wikipedia-1.4.0
C:\Users\diego>
```

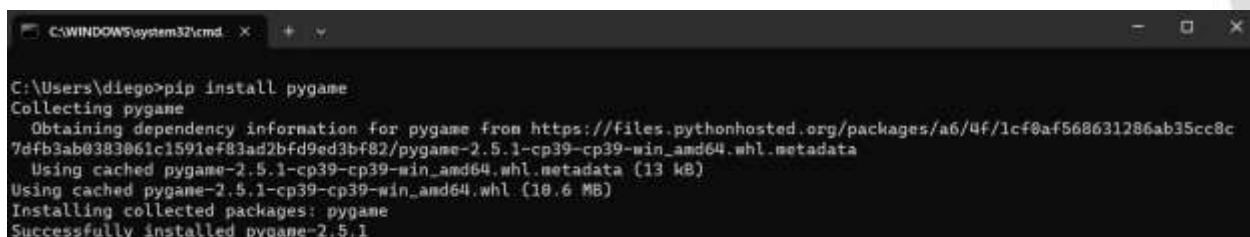
- **keyboard y pygame - Acción Asistente Virtual - Establecer una Alarma:** La habilidad de establecer una alarma cuando se le ordene al asistente virtual por medio de la palabra “alarma” se proporciona al ejecutar los siguientes comandos en la consola CMD de Windows, que realizan la instalación de las librerías keyboard y pygame. También se utiliza la librería datetime, pero esa no se debe instalar, ya viene incluida con Python.

```
pip install keyboard
```



```
C:\WINDOWS\system32\cmd. X + v
C:\Users\diego>pip install keyboard
Collecting keyboard
  Downloading keyboard-0.13.5-py3-none-any.whl (58 kB)
  Installing collected packages: keyboard
Successfully installed keyboard-0.13.5
```

```
pip install pygame
```



```
C:\WINDOWS\system32\cmd. X + v
C:\Users\diego>pip install pygame
Collecting pygame
  Obtaining dependency information for pygame from https://files.pythonhosted.org/packages/a6/4f/1cf0af568631286ab35cc8c7dfb3ab0383061c1591ef83ad2bfd9ed3bf82/pygame-2.5.1-cp39-cp39-win_amd64.whl.metadata
  Using cached pygame-2.5.1-cp39-cp39-win_amd64.whl.metadata (13 kB)
  Using cached pygame-2.5.1-cp39-cp39-win_amd64.whl (10.6 MB)
Installing collected packages: pygame
Successfully installed pygame-2.5.1
```



- **pyautogui - Acción Asistente Virtual - Mandar Mensaje por WhatsApp:** La habilidad de mandar un mensaje por WhatsApp cuando se le ordene al asistente virtual por medio de la palabra “WhatsApp” se proporciona al ejecutar el siguiente comando en la consola CMD de Windows, que realiza la instalación de la librería pyautogui. También se utilizan las librerías time y webbrowser, pero estas no se deben instalar, ya vienen incluidas con Python.

```
pip install pyautogui
```

```
C:\WINDOWS\system32\cmd. X + v
C:\Users\diego>pip install pyautogui
Collecting pyautogui
  Using cached PyAutoGUI-0.9.54-py3-none-any.whl
Requirement already satisfied: pynsbox in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyautogui) (1.0.9)
Requirement already satisfied: pytweneing>=1.0.4 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyautogui) (1.0.7)
Requirement already satisfied: pyscreeze>=0.1.21 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyautogui) (0.1.29)
Requirement already satisfied: pygetwindow>=0.0.5 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyautogui) (0.0.9)
Requirement already satisfied: mouseinfo in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyautogui) (0.1.3)
Requirement already satisfied: pyrect in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pygetwindow>=0.0.5->pyautogui) (0.2.0)
Requirement already satisfied: pyscreenshot in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyscreeze>=0.1.21->pyautogui) (3.1)
Requirement already satisfied: Pillow>=9.2.0 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyscreeze>=0.1.21->pyautogui) (10.0.0)
Requirement already satisfied: pyperclip in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from mouseinfo->pyautogui) (1.8.2)
Requirement already satisfied: EasyProcess in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyscreenshot->pyscreeze>=0.1.21->pyautogui) (1.1)
Requirement already satisfied: entrypoint2 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyscreenshot->pyscreeze>=0.1.21->pyautogui) (1.1)
Requirement already satisfied: nss in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from pyscreenshot->pyscreeze>=0.1.21->pyautogui) (9.0.1)
Installing collected packages: pyautogui
Successfully installed pyautogui-0.9.54
```

Código Python: A.V. - Mark I: Hablar, YouTube, OpenAI, Alarma, Notas y WhatsApp

```
# -*- coding: utf-8 -*-

#En Python se introducen comentarios de una sola linea con el simbolo #.

#La primera línea de código incluida en este programa se conoce como declaración de codificación o codificación
#de caracteres. Al especificar utf-8 (caracteres Unicode) como la codificación, nos aseguramos de que el archivo
#pueda contener caracteres especiales, letras acentuadas y otros caracteres no ASCII sin problemas, garantizando
#que Python interprete correctamente esos caracteres y evite posibles errores de codificación.

#Se puede detener una ejecución con el comando [CTRL] + C puesto en consola, con el comando "cls" se borra su
#historial y en Visual Studio Code con el botón superior derecho de Play se corre el programa.

#Para comentar en Visual Studio Code varias líneas de código se debe pulsar:
#[CTRL] + K (VSCode queda a la espera). Después pulsa [CTRL] + C para comentar y [CTRL] + U para descomentar.

#IMPORTACIÓN DE LIBRERÍAS:

#REPRODUCIR VIDEO EN YOUTUBE:

#pywhatkit: Pywhatkit es una biblioteca de Python que permite enviar mensajes de texto a través de WhatsApp
#incluso si no estás en su lista de contactos, realizar búsquedas en la web, reproducir canciones en YouTube,
#realizar búsquedas en Wikipedia, ejecutar comandos en consola, etc.
```



```

import pywhatkit

#IMPORTACIÓN DE CLASES: Cuando se quiera importar una clase, el nombre de esta no puede empezar con un número,
#sino cuando la quiera importar obtendré un error y se va accediendo a las carpetas o también llamados paquetes
#en la programación orientada a objetos (POO), por medio de puntos:
# - Directorio normal:      carpeta1/carpeta2/carpeta3
# - Directorio paquetes:    carpeta1.carpeta2.carpeta3
#La parte del directorio se coloca después de la palabra reservada from y la clase a importar después de import.
from POO_AsistenteVirtualMarkI.oidoAsistente import EscucharMicrofono
from POO_AsistenteVirtualMarkI.vozAsistente import vozWindows
from POO_AsistenteVirtualMarkI.cerebroLangchainAsistente import cerebro_OpenAI
from POO_AsistenteVirtualMarkI.alarmaAsistente import widgetAlarma
from POO_AsistenteVirtualMarkI.notasAsistente import widgetEscribirNotas
from POO_AsistenteVirtualMarkI.whatsappAsistente import widgetWhatsApp

#IMPORTACIÓN DE LIBRERÍAS:
#IMPORTACIÓN DE LLAVE: Cuando se quiera utilizar una API que utiliza un key, por seguridad es de buenas
#prácticas declararla en un archivo externo, además cabe mencionar que el nombre de dicho archivo y constante no
#pueden empezar con un número, sino cuando la quiera importar obtendré un error y se va accediendo a sus carpetas
#por medio de puntos:
# - Directorio normal:      carpeta1/carpeta2/carpeta3
# - Directorio paquetes:    carpeta1.carpeta2.carpeta3
#La parte del directorio se coloca después de la palabra reservada import y posteriormente se manda a llamar sus
#variables o constantes de igual manera a través de un punto.
import API_Keys.Llaves_ChatGPT_Bard
#ChatGPT API key
ApiKey = API_Keys.Llaves_ChatGPT_Bard.LlaveChatGPT
contacto = API_Keys.Llaves_ChatGPT_Bard.telefonoDicer0

#__name__ == __main__: Método main, esta función es super importante ya que sirve para instanciar las clases del
#programa y ejecutar sus métodos, en python pueden existir varios métodos main en un solo programa, aunque no es
#una buena práctica.
if (__name__ == "__main__"):
    #OBJETOS DE LAS FUNCIONES DEL ASISTENTE VIRTUAL:
    oidoAsistenteVirtual = EscucharMicrofono() #Instancia de la clase propia EscucharMicrofono.
    vozAsistenteVirtual = vozWindows() #Instancia de la clase propia vozWindows.
    cerebroAsistenteVirtual = cerebro_OpenAI(ApiKey) #Instancia de la clase propia cerebro_OpenAI.
    alarmaAsistenteVirtual = widgetAlarma() #Instancia de la clase propia widgetAlarma.
    horaAlarma = None #Variable que guarda el estado de la alarma.
    notasAsistenteVirtual = widgetEscribirNotas() #Instancia de la clase propia widgetAlarma.
    whatsappAsistenteVirtual = widgetWhatsApp(contacto) #Instancia de la clase propia widgetWhatsApp.
    vozAsistenteVirtual.hablar("Quiubolas di0, soy TIMI Mark 1, tu asistente virtual, puedo responder tus " +
                               "preguntas con ChatGPT, reproducir canciones o videos en YouTube, programar " +
                               "alarmas, escribir en mi archivo de notas y mandar mensajes por WhatsApp." +
                               "En que te puedo ayudar?")

```



```

while True:
    try:
        #EscucharMicrofono.oidoAsistenteVirtual(): Método propio de la clase EscucharMicrofono que se encarga
        #de escuchar lo que dice el usuario, almacenar eso en un archivo temporal y luego pasárselo al modelo
        #Whisper para que lo transcriba a texto.
        preguntaUsuario = oidoAsistenteVirtual.oidoAsistenteVirtual()
        print(str(preguntaUsuario) + "\n\n")
        #Si se reconoce que el usuario dice bye Timmy, se rompe el bucle while y se termina la ejecución del
        #programa.
        if (preguntaUsuario == " adiós, timmy." or preguntaUsuario == " ¡bai, timi!" or preguntaUsuario == " bye, timmy."
or preguntaUsuario == " adiós, tími." or preguntaUsuario == " adiós, dimi." or preguntaUsuario == " adiós, teamy." or
preguntaUsuario == " ¡bye, timmy!" or preguntaUsuario == " adios timmy" or preguntaUsuario == " bye timmy"):
            vozAsistenteVirtual.hablar("Bye di0, un gusto haberte ayudado...")
            print("-----Bye di_cer0, un gusto haberte ayudado...-----")
            break

        #Si se reconoce que el usuario dice la palabra reproduce, el asistente virtual por medio de la
        #librería pywhatkit reproduce el video que el usuario dijo en YouTube.
        elif (("reproduce") in preguntaUsuario):
            #.replace().lower(): Lo que hace el método replace() es reemplazar todas las palabras que
            #aparezcan en un string en otra cadena específica y el método lower() sirve para convertir todas
            #las letras de una cadena en minúsculas, esto es importante hacerlo porque al reproducir una
            #canción o mandar una búsqueda a internet, siempre es mejor tener puras minúsculas.
            cancionYoutube = preguntaUsuario.replace('reproduce', '').lower()
            vozAsistenteVirtual.hablar("Ok di0, voy a reproducir el video que pediste en YouTube")
            #pywhatkit.playonyt(): El método playonyt() de la librería pywhatkit permite reproducir un
            #video de YouTube en el navegador web predeterminado del sistema.
            pywhatkit.playonyt(cancionYoutube)
            print("-----Ok di_cer0, ya reproducí la canción del artista que pediste en YouTube...-----")

        #Si se reconoce que el usuario dice la palabra alarma, el asistente virtual por medio de una
        #de la clase widgetAlarma programa una alarma en la hora que el usuario haya indicado.
        elif (("alarma") in preguntaUsuario):
            #widgetAlarma.programarAlarma(): Método de la clase propia widgetAlarma que ejecuta una alarma
            #que solamente puede ser detenida al presionar la tecla de SpaceBar.
            horaAlarma = alarmaAsistenteVirtual.programarAlarma(preguntaUsuario)
            vozAsistenteVirtual.hablar("Hola di0, alarma puesta a las " + horaAlarma)
            print("-----Ok di_cer0, ya he establecido la alarma que me dijiste...-----")

        #Si no se reconoce que el usuario diga ninguna de las palabras reservadas de arriba, el asistente
        #virtual por medio de la librería de langchain y openai responde la pregunta hecha por el usuario.
        else:
            #widgetAlarma.ejecutarAlarma(): Método de la clase propia widgetAlarma que ejecuta una alarma
            #que solamente puede ser detenida al presionar la tecla de SpaceBar.

```




```

        respuestaAsistenteVirtual, respuestaArchivo, respuestaMensaje =
cerebroAsistenteVirtual.preguntarChatbot(preguntaUsuario)

        vozAsistenteVirtual.hablar(respuestaAsistenteVirtual)
        print(respuestaAsistenteVirtual)

        #Si se reconoce que el usuario dice la palabra escribe, el asistente virtual por medio del
        #método open() de Python escribe lo que el usuario dijo en un archivo txt.
        if (("escribe" or "guarda") in preguntaUsuario):
            vozAsistenteVirtual.hablar("Ok di0, voy a guardar la respuesta que te di en mi archivo de notas.")
            respuestaNotas = notasAsistenteVirtual.escribirNotaTxt(respuestaArchivo)
            vozAsistenteVirtual.hablar(respuestaNotas)

        if (("mensaje" or "Mensaje" or "whatsapp" or "Whatsapp") in preguntaUsuario):
            vozAsistenteVirtual.hablar("Ok di0, voy a mandarte mi respuesta por mensaje a tu whatsapp.")
            respuestaWhatsApp = whatsappAsistenteVirtual.mandarMensaje(respuestaMensaje)
            vozAsistenteVirtual.hablar(respuestaWhatsApp)

        #Condicional que está checando si la variable horaAlarma ya tiene asignada una hora de alarma.
        if (horaAlarma != None):
            respuestaAlarma = alarmaAsistenteVirtual.sonarAlarma(horaAlarma)
            vozAsistenteVirtual.hablar(respuestaAlarma)

        except Exception as errorEjecucion:
            vozAsistenteVirtual.hablar("Lo siento, no escuché bien lo que dijiste debido a este error...")
            print(errorEjecucion)
            continue

#El límite de la duración de audio que puede procesar T.I.M.M.Y. en las instrucciones que se le manda es de 15
#segundos y el Mark I en sus respuestas tiene un delay de:
# - tiny: Delay de 2 segundos cuando le mando un comando hablado usando el comando.
# - base: Delay de 5 segundos cuando le mando un comando hablado usando el comando.
# - small: Delay de 10 segundos cuando le mando un comando hablado usando el comando.

```

Clase: oidoAsistente

```

#IMPORTACIÓN DE LIBRERÍAS:
import io          #io: Librería que permite la manipulación de los archivos y carpetas de nuestro ordenador.
#AudioSegment: Clase de la librería pydub que permite tomar el audio que perciba el micrófono del ordenador y
#luego que eso se pueda convertir en un archivo mp3 o wav.
import pydub       #pydub: Librería para procesar y exportar a un archivo datos de audio.
import speech_recognition as sr #sr: Librería que permite recabar audio del micrófono en tiempo real.
import whisper      #whisper: Librería de OpenAI que permite hacer la transcripción de audio o video a texto.
import tempfile     #tempfile: Librería que crea y maneja archivos temporales.
import os           #os: Librería que permite acceder a funciones y métodos relacionados con el sistema operativo.

#CREACIÓN DE ARCHIVO TEMPORAL:
#tempfile.mkdtemp(): El método mkdtemp() crea un directorio temporal en la carpeta predeterminada del sistema
#operativo, la cual es llamada temp y es una ubicación en el sistema operativo donde se almacenan archivos
#temporales generados por diversas aplicaciones y procesos.

```



```

archivoTemporal = tempfile.mkdtemp()

#os.path.join(): El método join() se utiliza para unir varios componentes de ruta (variables y constantes) en
#una sola ruta. Esto se utilizará para crear el archivo wav o mp3 que guardará un pedazo de la instrucción del
#usuario. Cabe mencionar que WAV es un formato de archivo de audio sin pérdidas, lo que significa que todos los
#datos de audio originales se conservan. Esto hace que los archivos WAV sean de mayor calidad que los archivos
#MP3, pero también los hace más grandes y pesados.
rutaArchivo = os.path.join(archivoTemporal, 'audioTemporal.wav')
print("Esta es la ruta del archivo temporal de las instrucciones recibidas en audio:\n" + str(rutaArchivo))

#RECONOCIMIENTO DE LAS INSTRUCCIONES DADAS AL USUARIO:
#speech_recognition.Recognizer(): El objeto Recognizer() se utiliza para crear un reconocedor de voz, el cual
#puede utilizarse para reconocer instrucciones dadas al asistente virtual por medio del habla humana.
listenerInstrucciones = sr.Recognizer()

#EscucharMicrofono: Clase propia para escuchar las instrucciones del usuario por medio del micrófono y almacenar
#eso en un archivo temporal, luego se transcribirá ese archivo de audio a texto a través del modelo Whisper y
#finalmente el texto resultante será transformado a minúsculas para que sea interpretado por el programa.
class EscucharMicrofono:

    #POO: En Python cuando al nombre de una función se le ponen dos guiones bajos antes de su nombre es porque
    #se está refiriendo a un método privado, es una buena práctica de sintaxis.

    #__escuchaInstrucciones(textoAsistente): Función propia y con modificador de acceso privado que se encarga
    #de escuchar y reconocer el audio por medio del micrófono del ordenador las palabras dichas por el usuario,
    #esto se tarda un poco en cargar la primera vez que se corre el programa debido a que el método
    #speech_recognition.Recognizer().listen() necesita un tiempo para inicializarse y configurarse; si se está
    #utilizando una computadora con poca potencia o micrófonos de baja calidad, es posible que se necesite
    #esperar más tiempo.

    def __escuchaInstrucciones(self):

        #MANEJO DE EXCEPCIONES: Es una parte de código que se conforma de dos partes, try y except:

        # - Primero se ejecuta el código que haya dentro del try y si es que llegara a ocurrir una excepción
        #   durante su ejecución, el programa brinca al código del except.

        # - En la parte de código donde se encuentra la palabra reservada except, se ejecuta cierta acción
        #   cuando ocurra el error esperado.

        #Se utiliza esta arquitectura de código cuando se quiera efectuar una acción donde se espera que pueda
        #ocurrir un error durante su ejecución.

        try:

            #with as source: La instrucción with se utiliza para definir una variable que tiene asignada un
            #método o recurso específico, el cual puede ser un archivo, una conexión a una base de datos, la
            #cámara, micrófono o cualquier otro objeto que requiera ser cerrado. La palabra clave as se utiliza
            #para asignar dicho recurso a una variable que puede utilizarse para acceder al recurso dentro del
            #bloque with.

            #Dentro del manejo de excepciones la instrucción with as source: asegura que se cierre el recurso,
            #incluso si se produce un error dentro del bloque try.

            #speech_recognition.Microphone(): El objeto Microphone puede utilizarse para grabar audio.

            with sr.Microphone() as microfono:

                print("\n\n-----Hola soy T.I.M.M.Y. Mark I tu asistente virtual, en que te puedo ayudar?...-----")

```




```

#speech_recognition.Recognizer().adjust_for_ambient_noise(): El método adjust_for_ambient_noise()
#sirve para aplicar un filtro a la señal de sonido que quite el ruido de fondo recibido en su
#parámetro.

listenerInstrucciones.adjust_for_ambient_noise(microfono)
print("Quitando ruido de fondo... ya puedes hablar.")

#speech_recognition.Recognizer().listen(): El método listen() sirve para poder escuchar de una
#fuente de audio en tiempo real y convertirlo a texto, para ello primero se tuvo que haber
#instanciado la clase Recognizer.

textoInstruccionUsuario = listenerInstrucciones.listen(microfono)
print("Microfono encendido y escuchando...")

#io.BytesIO(): Método que sirve para crear un flujo de bytes en memoria que puede utilizarse
#para almacenar datos binarios, como imágenes, audio y video.

#speech_recognition.Recognizer().listen().get_wav_data(): El método get_wav_data() se utiliza
#para almacenar el texto recibido en el micrófono del ordenador en un flujo de datos binarios
#que se puedan guardar en un archivo de audio con extensión wav.

datosAudio = io.BytesIO(textoInstruccionUsuario.get_wav_data())

#pydub.AudioSegment().from_file(): El método from_file() perteneciente a la clase AudioSegment
#se utiliza para cargar solamente los datos de audio donde se escuche la voz del usuario,
#logrando así que no se haga nada cuando el usuario está en silencio.

archivoAudio = pydub.AudioSegment.from_file(datosAudio)
print("Mandando texto del microfono a un archivo de audio wav...")

#pydub.AudioSegment().from_file().export(path, file_type): Lo que hace el método export() es
#exportar un segmento de audio a un archivo.

archivoAudio.export(rutaArchivo, format = 'wav')
print("Texto del microfono guardado en un archivo de audio temporal tipo wav...")

#Para identificar el tipo de excepción que ha ocurrido y utilizarlo en la instrucción except, se puede
#utilizar la clase Exception, que es una clase incorporada en Python utilizada para describir todos los
#tipos de excepciones, luego de colocar el nombre de la clase Exception se usa la palabra reservada "as"
#seguida de un nombre de variable, esto nos permitirá acceder a la instancia de la excepción y
#utilizarla dentro del except.

except Exception as error:
    print(error)

#Se retorna la ruta del archivo de la función porque de ahí extraerá el archivo de audio el modelo
#Whisper para procesarlo.

return rutaArchivo

#__transcripcionWhisper(): Función propia y con modificador de acceso privado que se encarga de recibir el
#archivo de audio recortado que contiene la instrucción mandada al asistente virtual para pasársela al
#modelo de Whisper para que la transcriba a texto.

def __transcripcionWhisper(self, audioRecortado):
    #whisper.load_model(): Con el método load_model() se pueden cargar los diferentes modelos de lenguaje de
    #la librería whisper, si se elige un modelo muy simple, se tendrá un peor desempeño, pero mientras más
    #complejo sea el modelo, más se tardará en procesar el audio, a esto dentro de la librería se le llama
    #Parameter y Relative Speed, donde a un mayor número de parámetros, se tendrá mayor capacidad de
    #aprendizaje en el modelo y la velocidad relativa indica la rapidez con la que el modelo puede generar

```



```

#texto, esta se calcula dividiendo el tiempo que tarda el modelo en generar una palabra entre el número
#de palabras que genera. Las características de los modelos disponibles son las siguientes:
# - tiny: Parameter = 39M, Memoria que consume = 1GB y Relative Speed = 32x.
# - base: Parameter = 74M, Memoria que consume = 1GB y Relative Speed = 16x.
# - small: Parameter = 244M, Memoria que consume = 2GB y Relative Speed = 6x.
# - medium: Parameter = 769M, Memoria que consume = 5GB y Relative Speed = 2x.
# - large: Parameter = 1550M, Memoria que consume = 10GB y Relative Speed = 1x.
#Cuando el programa sea ejecutado por primera vez empezará a descargar el modelo para poderlo utilizar,
#este proceso tarda un poco en terminar, pero después identificará por sí solo el lenguaje y las
#palabras dichas en el audio, no importando si este tenga ruido o si es de alguna canción con
#instrumentos detrás.
Modelo = whisper.load_model("base")

#whisper.load_model().transcribe(): El método transcribe() sirve para escuchar el audio de un archivo y
#transcribirlo a texto a través del modelo elegido con el método load_model(). El proceso de carga tarda
#un poco y el tiempo que el modelo se tarda en procesar el audio depende de su duración. Los parámetros
#que recibe el método son la ubicación del archivo de audio que quiere transcribir y el parámetro fp16
#que indica si el cálculo se realizará en formato de punto flotante de 16 bits (FP16), pero como en
#algunas CPUs no se admite esta funcionalidad, se coloca como False, por lo que se está utilizando punto
#flotante de 32 bits (FP32) en su lugar. Además cabe mencionar que el audio escuchado por el método será
#recortado en cachos de 30 segundos y estos serán igualmente divididos en pedazos.
TranscripcionAudio_a_Texto = Modelo.transcribe(audioRecortado, language = "spanish", fp16 = False)
print("Interpretando texto guardado en un archivo temporal de audio wav con el modelo Whisper...\t")
#whisper.load_model().transcribe()["text"]: Texto traducido por Whisper de un archivo de audio.
return TranscripcionAudio_a_Texto["text"]

#oidoAsistenteVirtual(): Método con modificador de acceso público que permite ejecutar los métodos privados
#__transcripcionWhisper() para transcribir un archivo de audio a texto y __escuchaInstrucciones() para
#escuchar las instrucciones del usuario por medio del micrófono y almacenar eso en un archivo temporal. El
#resultado del texto escuchado del usuario será transformado a minúsculas aplicándole el método lower().
#self: La instrucción self se utiliza para hacer referencia al objeto que se está manipulando cuando se instancia
#la clase. Por eso es que a través de la misma nomenclatura de un punto se accede a los distintos atributos y/o
#métodos con un objeto desde fuera de la clase.
def oidoAsistenteVirtual(self):
    return self.__transcripcionWhisper(self.__escuchaInstrucciones()).lower()

```

Clase: vozAsistente

```

#IMPORTACIÓN DE LIBRERÍAS:
import pyttsx3 #pyttsx3: Biblioteca que sirve para generar una voz disponible en el sistema operativo.

#VOZ UTILIZADA POR EL ASISTENTE VIRTUAL:
#pyttsx3.init(): El método pyttsx3.init() se utiliza para inicializar el motor de texto a voz, el cual dará la
#habilidad de hablar al asistente virtual, convirtiendo el texto retornado por el modelo de Whisper en audio que
#salga de la computadora.
motorVoz = pyttsx3.init()

```

```

#pyttsx3.init().getProperty("voices"): El método getProperty() se utiliza para obtener una lista de las voces
#disponibles en el motor de texto a voz incluidos en el sistema operativo del ordenador.
voz_AsistenteVirtual = motorVoz.getProperty("voices")
#Bucle for para mostrar todas las opciones de sintetizadores de voz disponibles en el sistema operativo, el
#número de opciones dependerá de los lenguajes que pueda manejar el sistema operativo de la computadora que
#digan text to speech y se pueden ver al ingresar a la opción de: Windows -> Configuración -> Hora e Idioma ->
#Idioma y Región -> Idioma -> Idiomas que diga texto a voz. Si se quiere se podría agregar más idiomas.
for i in range(len(voz_AsistenteVirtual)):
    print(voz_AsistenteVirtual[i])

#pyttsx3.init().setProperty(): Este método toma dos parámetros y no devuelve ningún valor porque indica las
#siguientes características del sintetizador de voz del asistente virtual:
# - name: El nombre de la propiedad que se desea establecer.
#   - rate: Indica la velocidad de la voz en palabras por minuto.
#   - volume: El volumen de la voz, de 0 a 1.
#   - voices: Lista de todas las voces disponibles.
#   - voice: La voz que se utilizará para hablar.
#   - langauge: El idioma que se utilizará para hablar.
#   - engine: Controlador de voz que se utilizará.
#   - debug: Variable booleana que indica si se debe habilitar el modo de depuración o no.
# - value: El valor de la propiedad que se desea establecer.
motorVoz.setProperty("rate", 190)
motorVoz.setProperty("voice", voz_AsistenteVirtual[3].id)

#vozWindows: Clase propia que utiliza una de las voces incluidas en los lenguajes descargados en el sistema
#operativo Windows para lograr así que el asistente virtual hable.
class vozWindows:
    #hablar(textoAsistente): Función propia que permite al asistente virtual hablar por medio de la bocina del
    #ordenador.
    def hablar(self, textoAsistente):
        #pyttsx3.init().say(): El método say() se utiliza para convertir texto en audio y emitirlo a través del
        #altavoz del sistema.
        motorVoz.say(textoAsistente)
        #pyttsx3.init().runAndWait(): El método runAndWait() bloquea la ejecución del programa hasta que el motor
        #de texto a voz haya terminado de hablar.
        motorVoz.runAndWait()

```

Clase: cerebroLangchainAsistente

```

# -*- coding: utf-8 -*-

#En Python se introducen comentarios de una sola línea con el símbolo #.
#La primera línea de código incluida en este programa se conoce como declaración de codificación o codificación
#de caracteres. Al especificar utf-8 (caracteres Unicode) como la codificación, nos aseguramos de que el archivo
#pueda contener caracteres especiales, letras acentuadas y otros caracteres no ASCII sin problemas, garantizando
#que Python interprete correctamente esos caracteres y evite posibles errores de codificación.

```

```

#Se puede detener una ejecución con el comando [CTRL] + C puesto en consola, con el comando "cls" se borra su
#historial y en Visual Studio Code con el botón superior derecho de Play se corre el programa.
#Para comentar en Visual Studio Code varias líneas de código se debe pulsar:
#[CTRL] + K (VSCode queda a la espera). Después pulsa [CTRL] + C para comentar y [CTRL] + U para descomentar.

#IMPORTACIÓN DE LIBRERÍAS:

#1.-MODELOS (Models): El modelo se refiere a la red neuronal que se va a utilizar para procesar el texto de entrada y
#generar una respuesta, los Large Language Model (LLM) responden preguntas sin guardar un historial, mientras que los
#Chats sí guardan las preguntas y respuestas realizadas para crear una conversación. Existen varios modelos dentro de
#una misma compañía, por ejemplo, OpenAI cuenta con gpt3, gpt4, gpt3.5 turbo, etc.
print("-----LANGCHAIN-----")
print("-----1.-MODELOS-----")
#ChatOpenAI: Clase de la librería langchain que permite utilizar el modelo de chat (ChatGPT) de OpenAI con Python, este
#puede contestar preguntas adoptando un rol y guardar un historial durante la conversación.
from langchain.chat_models import ChatOpenAI    #ChatOpenAI: Modelo de Chat.

#2.-PROMPTS: Es el texto que se le envía al modelo para generar una respuesta y en este es donde se utilizan las
#técnicas de Prompt Engineering, para ello la librería LangChain cuenta con diferentes clases que permiten utilizar
#dichas técnicas, dependiendo de si se está mandando el Prompt a un LLM o a un Chat.
print("-----2.-PROMPTS-----")
#PromptTemplate: Clase de la librería langchain que permite mandar instrucciones o preguntas personalizadas a un modelo
#LLM (Large Language Model) previamente invocado con Python, que no guarda un historial.
from langchain import PromptTemplate            #PromptTemplate: Pregunta mandada a un modelo LLM.
#ChatPromptTemplate: Clase de la librería langchain que permite mandar instrucciones o preguntas personalizadas a un
#modelo de Chat, este puede contestar preguntas adoptando un rol a través de las siguientes clases:
# - SystemMessagePromptTemplate: Con esta clase se indica el rol que interpretará ChatGPT al responder las preguntas
#   del usuario.
# - HumanMessagePromptTemplate: Con esta clase se representa el rol del usuario que manda preguntas a ChatGPT.
from langchain.prompts import ChatPromptTemplate #ChatPromptTemplate: Instrucciones mandadas a un modelo de chat.
from langchain.prompts import SystemMessagePromptTemplate, HumanMessagePromptTemplate

#4.-CADENAS (Chains): Con esta herramienta se permite enlazar un modelo con un Prompt, también con ella se pueden
#conectar varios modelos entre sí, hasta cuando son de distintos tipos, permitiéndonos así realizar varias iteraciones
#entre modelos durante una consulta para obtener un mejor procesamiento final de los datos cuando este se busca aplicar
#a tareas muy complejas.
print("-----4.-CADENAS-----")
#PROCESAMIENTO DE LLM: Permite encadenar un prompt con varios modelos de forma secuencial, uniendo así varias cadenas.
# - TransformChain: Con esta clase se implementa una cadena de transformación, que se aplica a una entrada para
#   producir una salida con un formato personalizado. Las transformaciones pueden ser representadas por cualquier
#   función que tome una secuencia como entrada y devuelva una secuencia como salida.
#Por lo tanto, para que una cadena TransformChain funcione, se debe declarar una función propia que cambie el formato
#de la salida de otra cadena.
from langchain.chains import TransformChain    #TransformChain: Librería que crea una cadena de cadenas.

class cerebro_OpenAI:

```



```

#CONSTRUCTOR O INICIALIZADOR DE LA CLASE: En él se declaran los parámetros que recibe la clase, que además se
#utilizarán en los demás métodos, estos a fuerza deben tener un valor.

#self: La instrucción self se utiliza para hacer referencia al objeto que se está manipulando cuando se instancia
#la clase. Por eso es que a través de la misma nomenclatura de un punto se accede a los distintos atributos y/o
#métodos con un objeto desde fuera de la clase.

def __init__(self, parametro_de_la_clase):
    #ChatGPT API key
    self.LlaveApi = parametro_de_la_clase

#POO: En Python cuando al nombre de una función se le ponen dos guiones bajos antes de su nombre es porque
#se está refiriendo a un método privado, es una buena práctica de sintaxis.

#__ChatbotLangchain(preguntaUsuario): Función propia y con modificador de acceso privado que se encarga de recibir
#el texto de la pregunta realizada al modelo de Chat de OpenAI para contestar una pregunta según un rol asignado.
def __ChatbotLangchain(self, preguntaUsuario):
    #ChatOpenAI(): En el constructor de la clase ChatOpenAI del paquete chat_models de la librería langchain se
    #indica:
    # - model_name: Parámetro que indica el modelo que se quiere utilizar, en este caso se utilizará gpt-3.5-turbo
    #   que pertenece a GPT-3.5.
    # - openai_api_key: Con este parámetro se proporciona la API key, que por buenas prácticas debe provenir de
    #   otro archivo.
    # - prompt_length: Longitud del prompt.
    # - max_tokens: Número máximo de tokens que se pueden generar.
    # - stop_token: El token de parada.
    # - temperature: La temperatura es un valor entre 0 y 1 que indica la creatividad con la que contesta el LLM,
    #   si es demasiado grande, puede responder con algo totalmente aleatorio y si es muy bajo responderá lo mismo
    #   siempre, función que podría ser deseada cuando por ejemplo se contestan problemas matemáticos.
    #Todos los modelos disponibles para usarse con OpenAI estan enlistados en el siguiente enlace y cada uno es
    #mejor en ciertas funciones que el otro:
    #https://platform.openai.com/docs/models
    openaiChatGPT = ChatOpenAI(model_name = "gpt-3.5-turbo", openai_api_key = self.LlaveApi, temperature = 0.4)#Chat.

#SYSTEM - ROL DEL CHAT AL RESPONDER PREGUNTAS DEL USUARIO: Para ello se utiliza un objeto PromptTemplate.
#PromptTemplate(): En el constructor de la clase PromptTemplate perteneciente a la librería langchain se
#indica:
# - template: Parámetro que indica la pregunta del prompt.
# - input_variables: Indica a través de una lista todos los nombres de las variables incluidas en la plantilla
#   del prompt, que se declararon dentro del template entre llaves {}.
plantillaPromptSistema = PromptTemplate(
    template = "Tu nombre es Timmy, eres una asistente virtual carismática, burlona y experta en chistes de Chihuahueros."
    "Puedes contestar las preguntas del usuario espontáneamente de forma sarcástica y otras veces de forma"
    "seria y lógica para resolver sus problemas. El nombre del usuario es di0."
    "Tu como asistente virtual puedes establecer alarmas, escribir en tu archivo NotasTimmy.txt, reproducir"
    "videos en YouTube y mandar mensajes por Whatsapp.",

```



```

        input_variabels = []
    )
    #SystemMessagePromptTemplate(): Esta clase recibe como parámetro un objeto PromptTemplate, que previamente ya
    #tiene diseñado el template que se mandará en el Prompt, indicándole al Chat el rol que está interpretando al
    #responder.
    promptSistema = SystemMessagePromptTemplate(prompt = plantillaPromptSistema)

    #HUMAN - PREGUNTAS QUE EL USUARIO LE HACE AL MODELO: Para ello se utiliza un objeto PromptTemplate.
    plantillaPromptHumano = PromptTemplate(
        template = "Con tu vasto conocimiento que te vuelve un genio contesta la siguiente pregunta del usuario:\n"
            "{pregunta}",
        input_variabels = ["pregunta"]
    )
    #HumanMessagePromptTemplate(): Esta clase recibe como parámetro un objeto PromptTemplate, que previamente ya
    #tiene diseñado el template de la pregunta que hace el usuario al chat.
    promptHumano = HumanMessagePromptTemplate(prompt = plantillaPromptHumano)

    #ChatPromptTemplate.from_messages(): Método que sirve para unificar los templates previamente creados para el
    #sistema (que le dice al modelo el rol que debe interpretar al responder mis preguntas), para el humano (que
    #indica tal cual la pregunta realizada por el usuario) y de la AI (que es un rol adoptado por el modelo para
    #guardar las preguntas y respuestas realizadas en un historial), creando así una conversación. El parámetro que
    #recibe el método es una lista que incluye todas las plantillas de Prompt mencionadas previamente.
    plantillaChatPrompt = ChatPromptTemplate.from_messages([promptSistema, promptHumano])
    #ChatPromptTemplate().format_prompt().to_messages(): Método que rellena las variables del template mandado al
    #Chat con valores de entrada para el prompt del sistema, del humano y de la AI, retornando una lista.
    promptMandadoChat = plantillaChatPrompt.format_prompt(pregunta = preguntaUsuario).to_messages()
    #str(): Método que convierte un número, lista, diccionario, etc. en un string para que pueda ser impreso en
    #consola.
    print("Personalidad de Chatbot:\n" + str(promptMandadoChat[0].content) + "\n\n" +
        "Pregunta hecha al Chatbot:\n" + str(promptMandadoChat[1].content) + "\n")
    #ChatOpenAI(ChatPromptTemplate().format().to_messages()): Prompt mandado al modelo de Chat.
    respuestaChat = openaiChatGPT(promptMandadoChat)
    #Del diccionario retornado, el key de content es el que contiene la respuesta de la pregunta.
    print("Respuesta del Chatbot:\n", respuestaChat.content + "\n\n")
    return respuestaChat.content

    #__formatoHablarRespuesta(respuestaChat): Función propia y con modificador de acceso privado que se encarga de
    #recibir la respuesta proporcionada por el modelo de Chat de OpenAI, para luego a través de una cadena de formato
    #llamada TranformChain(), quitar los saltos de línea del texto para que pueda hablar de forma fluida el asistente
    #virtual.
    def __formatoHablarRespuesta(self, respuestaChat):
        #Función propia que cambia el formato de cualquier salida proporcionada por un modelo de Chat o LLM.
        def eliminarSaltosDeLinea(entrada):
            #Función que intercambia los saltos de línea por espacios.
            texto = entrada["texto"] #Recibe una lista con un diccionario interno de key = texto.

```



```

        #lista.replace(): Método que reemplaza dentro de una lista un string por otro.
        return {"texto_limpio" : texto.replace("\n", " ")}

#TransformChain(): Objeto que recibe un prompt, cambia su formato de una forma personalizada y lo retorna en
#una variable nueva.

# - input_variabls: Indica a través de una lista los prompts de entrada.
# - output_variabls: Indica a través de una lista el nombre de la variable de salida ya con el formato
# deseado.
# - transform: Recibe el nombre de la función propia que transforma el formato de la variable de entrada.
cadenaTransformarFormato = TransformChain(input_variabls = ["texto"],
                                           output_variabls = ["texto_limpio"],
                                           transform = eliminarSaltosDeLinea)

transformChainHablar = cadenaTransformarFormato.run(respuestaChat)
return transformChainHablar

#__formatoMensajeRespuesta(respuestaChat): Función propia y con modificador de acceso privado que se encarga de
#recibir la respuesta proporcionada por el modelo de Chat de OpenAI, para luego a través de una cadena de formato
#llamada TransformChain() transformar la salida en una lista, con el fin de que el formato de la respuesta que se
#mandará por mensaje a WhatsApp se respete.
def __formatoMensajeRespuesta(self, respuestaChat):
    def listaSaltosDeLinea(entrada):
        texto = entrada["texto"]
        return {"texto_lista_mensaje" : texto.split("\n")}
    cadenaTransformarFormato = TransformChain(input_variabls = ["texto"],
                                              output_variabls = ["texto_lista_mensaje"],
                                              transform = listaSaltosDeLinea)

    transformChainMensaje = cadenaTransformarFormato.run(respuestaChat)
    return transformChainMensaje

#preguntarChatbot(preguntaUsuario): Método con modificador de acceso público que permite ejecutar los métodos
#privados __ChatbotLangchain(), __formatoHablarRespuesta y __formatoMensajeRespuesta para procesar la pregunta
#hecha por el usuario y responder en forma de lista que separa las líneas de texto de la respuesta y en forma de
#texto sin saltos de línea para que el asistente virtual pueda decir la respuesta y además la pueda mandar por
#medio de Whatsapp y que su formato se respete.
def preguntarChatbot(self, preguntaUsuario):
    respuestaArchivo = self.__ChatbotLangchain(preguntaUsuario) #Respuesta del método __ChatbotLangchain.
    respuestaHablar = self.__formatoHablarRespuesta(respuestaArchivo) #Formato de respuesta sin saltos de línea.
    respuestaMensaje = self.__formatoMensajeRespuesta(respuestaArchivo) #Formato de respuesta en forma de lista.
    #return variable_1, variable_2, ..., variable_n: Cuando después de un método return se tiene más de 1
    #variable, al ejecutar el método se deberán declarar dos variables que reciban cada resultado de forma
    #correspondiente a través de la siguiente sintaxis:
    # variable_1, variable_2, variable_n = objetoDeClase.métodoVariasVariables()
    return respuestaHablar, respuestaArchivo, respuestaMensaje

```



Clase: alarmaAsistente

```
#ESTABLECER ALARMA:

import datetime #datetime: Librería que proporciona método para trabajar con fechas y horas en Python.
#re: Librería que identifica patrones en expresiones regulares. Una expresión regular es un patrón de texto que se
#puede utilizar para buscar, reemplazar o extraer texto de una cadena.
import re #re: Librería que permite utilizar expresiones regulares para detectar patrones.
import keyboard #keyboard: Librería que permite detectar las pulsaciones de teclas en tu teclado
import pygame #pygame: librería de multimedia que permite mostrar gráficos, reproducir sonido, etc.

class widgetAlarma:
    def programarAlarma(self, preguntaUsuario):
        #.replace().lower(): Lo que hace el método replace() es reemplazar todas las palabras que aparezcan en un
        #string en otra cadena específica y el método lower() sirve para convertir todas las letras de una cadena en
        #minúsculas, esto es importante hacerlo porque al reproducir una canción o mandar una búsqueda a internet,
        #siempre es mejor tener puras minúsculas.
        instruccionHora = preguntaUsuario.replace(' alarma', '').lower()
        #strip(): Método utilizado para eliminar los caracteres especificados en una cadena, como espacios saltos de
        #línea (\n), tabuladores (\t), etc. Si no se le pasa ningún parámetro, eliminará los espacios del string.
        instruccionHora = instruccionHora.strip()
        print(instruccionHora)
        #re.findall(r): El método findall() utilizado para encontrar todas las ocurrencias de una expresión regular en
        #una cadena de texto (string) recibe 3 parámetros, la expresión regular que se quiere encontrar, la cadena de
        #caracteres donde se buscará y modificadores opcionales que se pueden utilizar para personalizar el
        #comportamiento de la búsqueda.
        # - pattern: Parámetro que indica la expresión regular utilizada para extraer partes específicas del string.
        # Las expresiones regulares más comunes son:
        # - Búsqueda de números en un string:
        # - \d: Expresión regular que busca un dígito del 0 al 9 en un string.
        # - \d+: Expresión regular que coincide con uno o más dígitos del 0 al 9.
        # - \d{2,4}: Expresión regular que coincide con una secuencia de dígitos de 2 al 4 específicamente,
        # osea 234.
        # - Búsqueda de letras en un string:
        # - \w: Expresión regular que busca una letra mayúscula o minúscula, un dígito del 0 al 9 o un guión
        # bajo.
        # - \w+: Expresión regular que coincide con uno o más caracteres de los descritos anteriormente.
        # - [A-Za-z]: Expresión regular que busca solo una letra mayúscula o minúscula.
        # - \b: Expresión regular que representa el inicio o final de una palabra en un string, usualmente se
        # incluyen dos para indicar que se debe buscar un patrón en cada palabra. Por ejemplo, si se declara la
        # expresión regular \bHola\b, esto extraerá todas las palabras Hola dentro de un string.
        # - Búsqueda de palabras en un string (grupos de captura):
        # - (patrón1|patrón2): Expresión regular que crea un grupo de captura para identificar uno o varios
        # patrones específicos.
        # - |: Compuerta OR utilizada para reconocer uno o varios patrones dentro de un grupo de captura.
        # - Búsqueda de espacios en blanco, puntos y paréntesis en un string:
        # - \s: Expresión regular que coincide con un carácter de espacio vacío, como un espacio, tabulador (\t),
```



```

#         salto de línea (\n), etc.
#
# - \.: Expresión regular que encuentra un punto en un string.
#
# - \(: Expresión regular que encuentra un paréntesis de apertura en un string.
#
# - \): Expresión regular que encuentra un paréntesis de cierre en un string.
#
# - Búsqueda de repeticiones en un string:
#
#   - *: Encuentra cero o más repeticiones de la expresión regular que tenga a la izquierda.
#
#   - +: Encuentra una o más repeticiones de la expresión regular que tenga a la izquierda.
#
#   - ?: Encuentra cero o una repetición de la expresión regular que tenga a la izquierda. Esto significa
#         que la expresión regular que la precede es opcional y puede aparecer una vez o no aparecer en
#         absoluto en la cadena que se está buscando.
#
#   - {2,4}: Coincide con 2, 3 o 4 repeticiones de la expresión regular que tenga a la izquierda.
#
# - Anclaje de patrones:
#
#   - ^: Cuando se coloca al principio de un patrón indica que la coincidencia debe encontrarse al comienzo
#         de la línea de texto. Por ejemplo, si se declara la expresión regular ^Hola, esto será cierto solo
#         cuando la palabra Hola se encuentre al inicio del string.
#
#   - $: Cuando se coloca al final de un patrón indica que la coincidencia debe encontrarse al final de la
#         línea de texto. Por ejemplo, si se declara la expresión regular $mundo, esto será cierto solo cuando
#         la palabra mundo se encuentre al final del string.
#
# - string: Palabra en la que se desea buscar patrones a través de la expresión regular especificada.
#
# - flags: Modificadores opcionales que personalizan el comportamiento de la búsqueda.
#
#   - re.IGNORECASE: Realiza la búsqueda sin distinción entre mayúsculas y minúsculas.
#
#   - re.MULTILINE: Permite que el patrón coincida con múltiples líneas en la cadena.
#
#   - re.DOTALL: Hace que el carácter . en el patrón coincida también con el carácter de nueva línea \n.
#Esta expresión regular busca números en un string.
numeroshora = re.findall(r"\d+", instruccionHora)
print(numeroshora)

#join(): Este método toma una lista de strings como argumento y los concatena, juntando cada string con el
#carácter especificado.
hora = ":".join(numeroshora)
print(hora)

#Para que el string que representa la hora pueda ser convertido a un objeto datetime con formato de 12
#horas, se debe indicar si la hora de este es pm o am, pero para que esto sea entendido por el método
#strptime(), se debe indicar en mayúsculas.
if("pm" in instruccionHora):
    hora += " PM"
else:
    hora += " AM"

#datetime.datetime.strptime(): Método de la clase datetime que toma una cadena que representa una fecha y
#hora en un formato específico y la convierte a un objeto datetime, para ello recibe las siguientes
#instrucciones en su segundo parámetro:
#
# - %H: Hora en formato de 24 horas (00-23).
#
# - %I: Hora en formato de 12 horas (01-12). Para esto se debe indicar si la hora es AM o PM.
#
# - %p: AM o PM (funciona con %I para indicar la parte de la tarde o la mañana).
#
# - %M: Minutos (00-59).

```



```

# - %S: Segundos (00-59).
# - %f: Microsegundos (000000-999999).
# - %z: Desplazamiento de la zona horaria UTC en formato ±HHMM o ±HH:MM.
hora = datetime.datetime.strptime(hora, "%I:%M %p") # Usa "%I" para el formato de 12 horas
#Conversión de objeto datetime con formato de 12 horas en formato de 24 horas. Esto se hace para que sea
#compatible con la hora actual retornada por medio del método datetime.datetime.now().strftime("%H:%M").
horaAlarma = hora.strftime("%H:%M")
print(horaAlarma)

#datetime.datetime.now().strftime(): Método que devuelve la fecha y hora actual en el formato
#YYYY-MM-DD HH:MM:SS, al cual se accede con el string "%Y-%m-%d %H:%M:%S".
print(datetime.datetime.now().strftime("%H:%M"))
return horaAlarma

def sonarAlarma(self, horaAlarma):
    respuestaAlarma = ""
    if (datetime.datetime.now().strftime("%H:%M") == horaAlarma):
        while True:
            print("-----Hola di_cer0, ya es la hora en la que me pediste que estableciera una alarma-----")
            #pygame.mixer.init(): El método init() inicializa el objeto mixer de la librería pygame
            #que sirve para reproducir sonidos con Python.
            pygame.mixer.init()
            #pygame.mixer.music.load(): El método load() carga un archivo de música con formato wav,
            #mp3 u ogg en la memoria para que pueda ser reproducido al ejecutar el método play().
            pygame.mixer.music.load("C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/0.-Archivos_Ejercicios_Python/Till I Collapse - Eminem.mp3")
            pygame.mixer.music.play()
            #keyboard.read_key(): Método que devuelve la tecla presionada por el usuario.
            if(keyboard.read_key() == "space"):
                #pygame.mixer.music.stop(): Método que detiene la música que se haya reproducido
                #anteriormente con la función play().
                pygame.mixer.music.stop()
                break
            horaAlarma = None
            respuestaAlarma = "Hola di0, alarma detenida"
        return respuestaAlarma

```

Clase: notasAsistente

```

#ABRIR Y GUARDAR NOTAS EN UN ARCHIVO TXT:
import os #os: Librería que permite acceder a funciones del sistema operativo, como abrir archivos.

class widgetEscribirNotas:
    def escribirNotaTxt(self, respuestaAsistenteVirtual):
        respuestaNotas = ""

```

```

filename = "C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/NotasTimmy.txt"

#MANEJO DE EXCEPCIONES: Es una parte de código que se conforma de dos partes, try y except:
# - Primero se ejecuta el código que haya dentro del try y si es que llegara a ocurrir una excepción
#   durante su ejecución, el programa brinca al código del except.
# - En la parte de código donde se encuentra la palabra reservada except, se ejecuta cierta acción
#   cuando ocurra el error esperado.
#Se utiliza esta arquitectura de código cuando se quiera efectuar una acción donde se espera que pueda
#ocurrir un error durante su ejecución.
try:

    #with as source: La instrucción with se utiliza para definir una variable que tiene asignada un
    #método o recurso específico, el cual puede ser un archivo, una conexión a una base de datos, la
    #cámara, micrófono o cualquier otro objeto que requiera ser cerrado. La palabra clave as se utiliza
    #para asignar dicho recurso a una variable que puede utilizarse para acceder al recurso dentro del
    #bloque with.

    #Dentro del manejo de excepciones la instrucción with as source: asegura que se cierre el recurso,
    #incluso si se produce un error dentro del bloque try.

    #open(): Método que sirve para abrir un archivo cualquiera y manejarlo por medio de Python.
    with open(filename, "w", encoding="utf-8") as f:

        f.write(respuestaAsistenteVirtual)

        #var_file_open.close(): Método para cerrar un archivo previamente abierto con el
        #método open(), es peligroso olvidar colocar este método, ya que la computadora lo
        #considerará como si nunca hubiera sido cerrado, por lo cual no podré volver a
        #abrirlo al dar clic sobre él.

        f.close()

        #os.system(): Método que permite ejecutar comandos del sistema operativo desde
        #Python.

        # - ls: Mostrar todas las carpetas del directorio.
        # - cd: Command directory sirve para moverme de directorio.
        # - start: Abrir o ejecutar un archivo.
        # - ejecutable + archivo: Con este comando que incluye un ejecutable y un archivo
        #   se indica al sistema operativo que abra cierto archivo usando un programa en
        #   específico.

        programa_para_abrir = "notepad.exe"
        os.system(f'{programa_para_abrir} "{filename}"')
except FileNotFoundError as errorArchivo:

    #open(): Método que sirve para abrir un archivo cualquiera, para ello es necesario
    #indicar dos parámetros, el primero se refiere a la ruta relativa o absoluta del archivo
    #previamente creado y la segunda indica qué es lo que se va a realizar con él, el
    #contenido del archivo se asigna a una variable.

    # - w: Sirve para escribir en un archivo, pero borrará la información que previamente
    #   contenía el archivo.
    # - a: Sirve para escribir en un archivo sin que se borre la info anterior del archivo,
    #   se llama append.

    file = open(filename, "w", encoding="utf-8")

```



```

file.write(respuestaAsistenteVirtual)
file.close()

#os.system(): Método que permite ejecutar comandos del sistema operativo desde Python.
# - ls: Mostrar todas las carpetas del directorio.
# - cd: Command directory sirve para moverme de directorio.
# - start: Abrir o ejecutar un archivo.
# - ejecutable + archivo: Con este comando que incluye un ejecutable y un archivo se
#   indica al sistema operativo que abra cierto archivo usando un programa en
#   específico.

programa_para_abrir = "notepad.exe"
os.system(f'{programa_para_abrir} "{filename}"')

print("-----Hola di_cer0, ya respondí tu pregunta y guardé la respuesta en mis notas...-----")
respuestaNotas = "Hola di0, ya escribí la respuesta que te dí en mi archivo de notas y luego te lo mostré."
return respuestaNotas

```

Clase: whatsappAsistente

```

#IMPORTACIÓN DE LIBRERÍAS:
import webbrowser #webbrowser: Librería que permite abrir y utilizar navegadores web en Python.
import pyautogui #pyautogui: Librería para controlar el mouse y teclado de la computadora con Python.
import time #time: Librería del manejo de tiempos, como retardos, contadores, etc.

class widgetWhatsApp:

    #CONSTRUCTOR O INICIALIZADOR DE LA CLASE: En él se declaran los parámetros que recibe la clase, que además se
    #utilizarán en los demás métodos, estos a fuerza deben tener un valor.

    #self: La instrucción self se utiliza para hacer referencia al objeto que se está manipulando cuando se instancia
    #la clase. Por eso es que a través de la misma nomenclatura de un punto se accede a los distintos atributos y/o
    #métodos con un objeto desde fuera de la clase.

    def __init__(self, parametro_de_la_clase):
        #ChatGPT API key

        self.contacto = parametro_de_la_clase

    def mandarMensaje(self, mensaje):
        respuestaWhatsApp = ""

        #Bucle for que recorre la respuesta del modelo de chat de OpenAI que viene en forma de lista para mandar cada
        #línea de texto en un mensaje diferente, ya que no es posible de forma gratuita mandar todo el texto en un
        #solo mensaje respetando los saltos de línea del texto.

        for i in range(len(mensaje)):

            #Solo en la primera línea de texto se abre el navegador y se espera que se habra bien para mandar el
            #primer mensaje.

            if(i == 0):

                #webbrowser.open(): Método que abre un navegador web en la dirección URL especificada. El truco que
                #se realiza para mandar mensajes de whatsapp es que la URL de su sitio permite ingresar el mensaje y
                #teléfono de la persona a la que se busca mandar el WhatsApp.

                webbrowser.open(f"https://web.whatsapp.com/send?phone={self.contacto}&text={mensaje[i]}")

```



```

#time.sleep(): Método que permite detener el programa cierto número de segundos.
time.sleep(8)

#pyautogui.press(): Método que simula presionar una tecla del teclado. El parámetro obligatorio que
#recibe es el nombre de la tecla, que puede ser una cadena o un valor ASCII. El método también acepta
#un parámetro opcional llamado interval, que especifica el intervalo de tiempo en segundos entre
#presionar la tecla y soltarla.

pyautogui.press("enter")

#Los demás mensajes simplemente se mandan al introducir cada línea de mensaje en la página de Whatsapp
#Web y presionar rápidamente la tecla de enter después de introducir cada línea del mensaje.
else:

    pyautogui.write(mensaje[i])
    pyautogui.press("enter")

#Finalmente se espera un tiempo y se cierra el navegador donde se encuentra abierta la pestaña.
pyautogui.write("-----MENSAJE DE T.I.M.Y. RECIBIDO CORRECTAMENTE-----")
pyautogui.press("enter")
time.sleep(2)

respuestaWhatsApp = "Hola di0, ya te mandé un Whats con mi respuesta a tu cel."
return respuestaWhatsApp

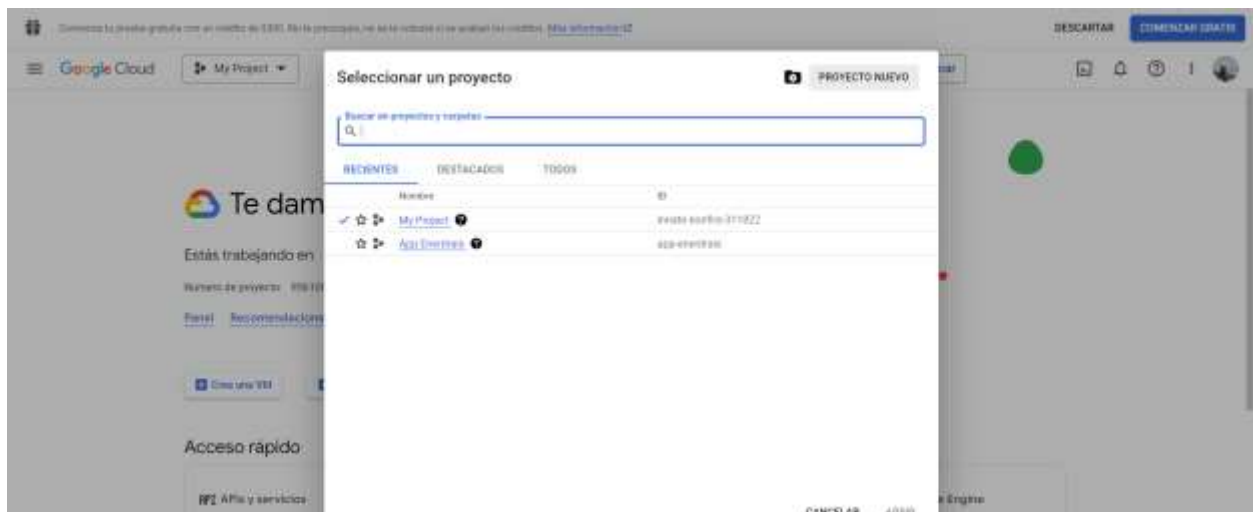
```

Asistente Virtual - Mark I: Google Calendar, Mapas, Visión Artificial y Machine Learning

API Google Cloud: Herramienta de Google Calendar con Python

Para utilizar la herramienta de Google Calendar a través de un código hecho con el lenguaje de programación Python se debe crear una API Key de Google Cloud, esto requiere que nos registremos a través de un correo de Gmail en el siguiente enlace, luego deberemos dar clic en el botón de My Project → Proyecto Nuevo:

<https://console.cloud.google.com/>



Posteriormente asignaremos un nombre a este nuevo proyecto y daremos clic en Crear:

Proyecto nuevo

Tienes 8 proyectos restantes en tu cuenta. Solicita un incremento o borra algunos proyectos. [Más información](#)

[MANEJAR COTIZAS](#)

Nombre del proyecto *

Asistente Virtual

ID de proyecto: asistente-virtual-218021. No se podrá cambiar más tarde. [EDITAR](#)

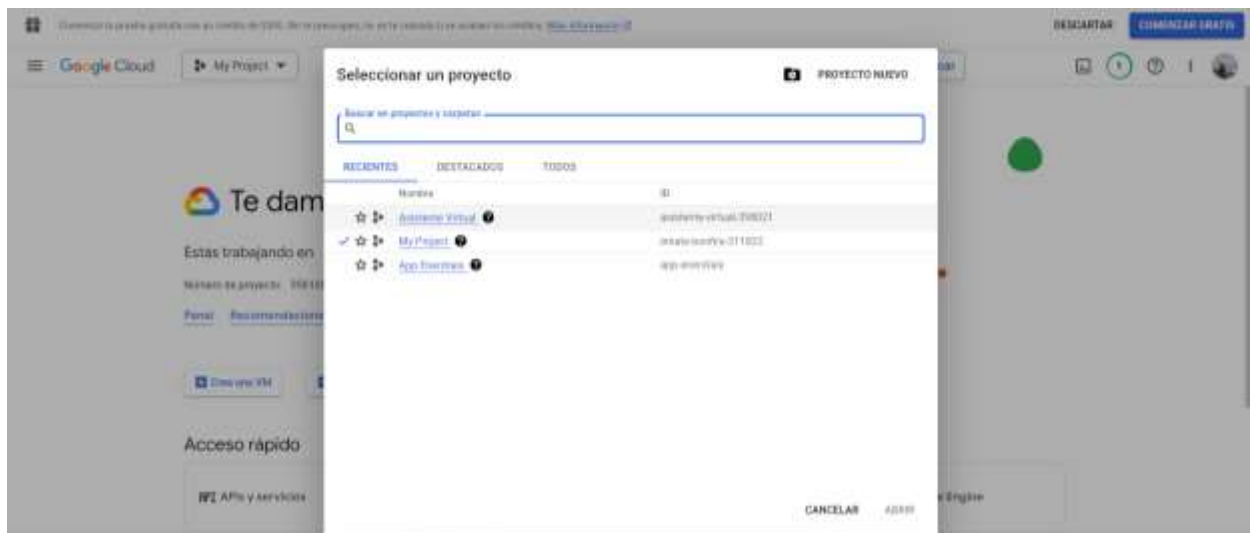
Ubicación *

Sin organización [EXPLORAR](#)

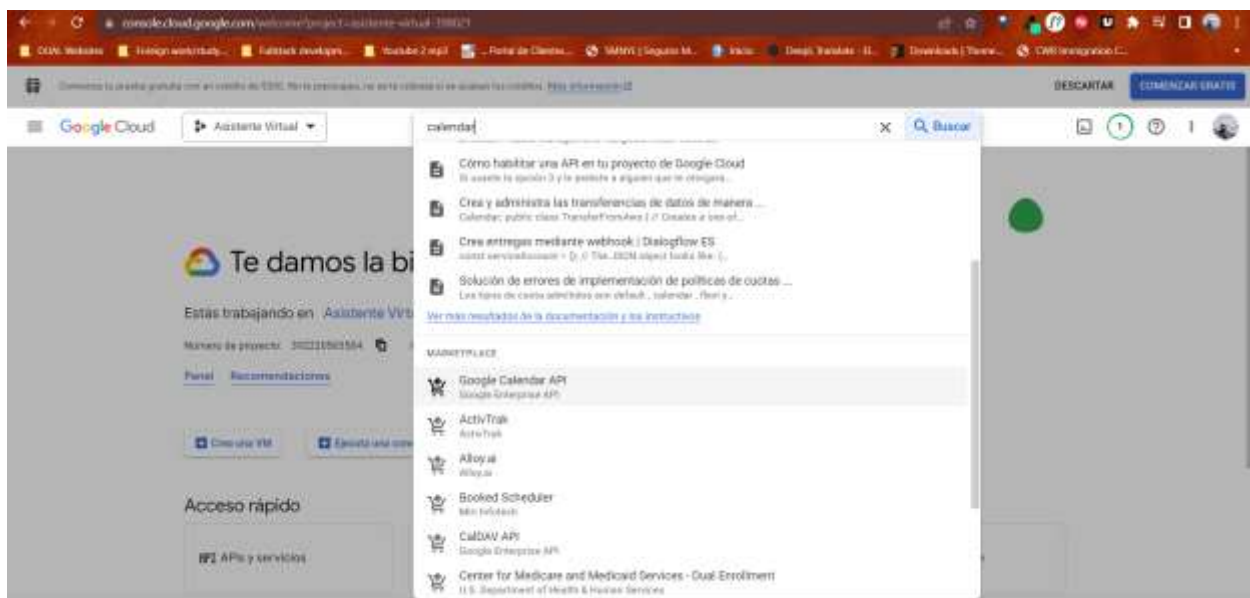
Organización o carpeta superior

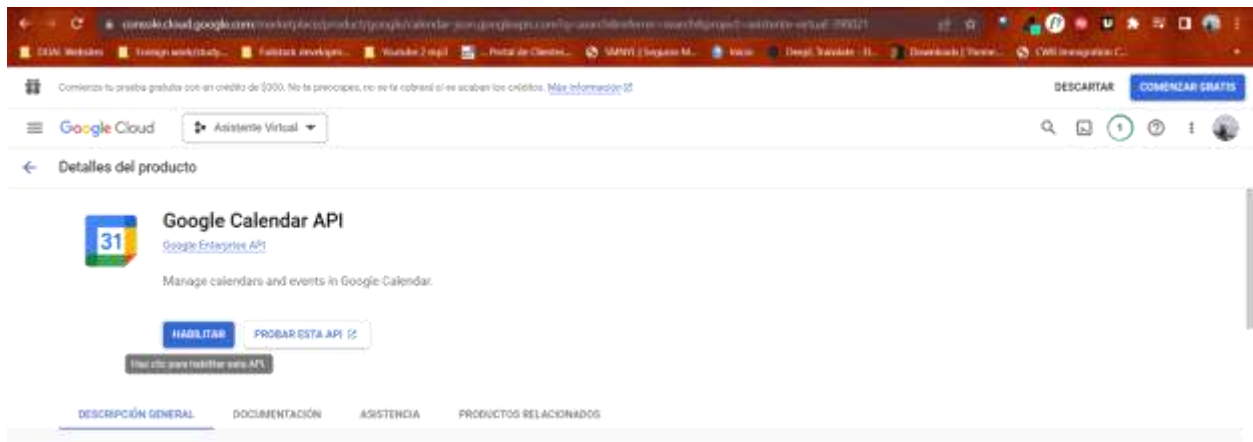
[CREAR](#) [CANCELAR](#)

Luego daremos de nuevo clic en la opción de My Project → Nombre Proyecto.

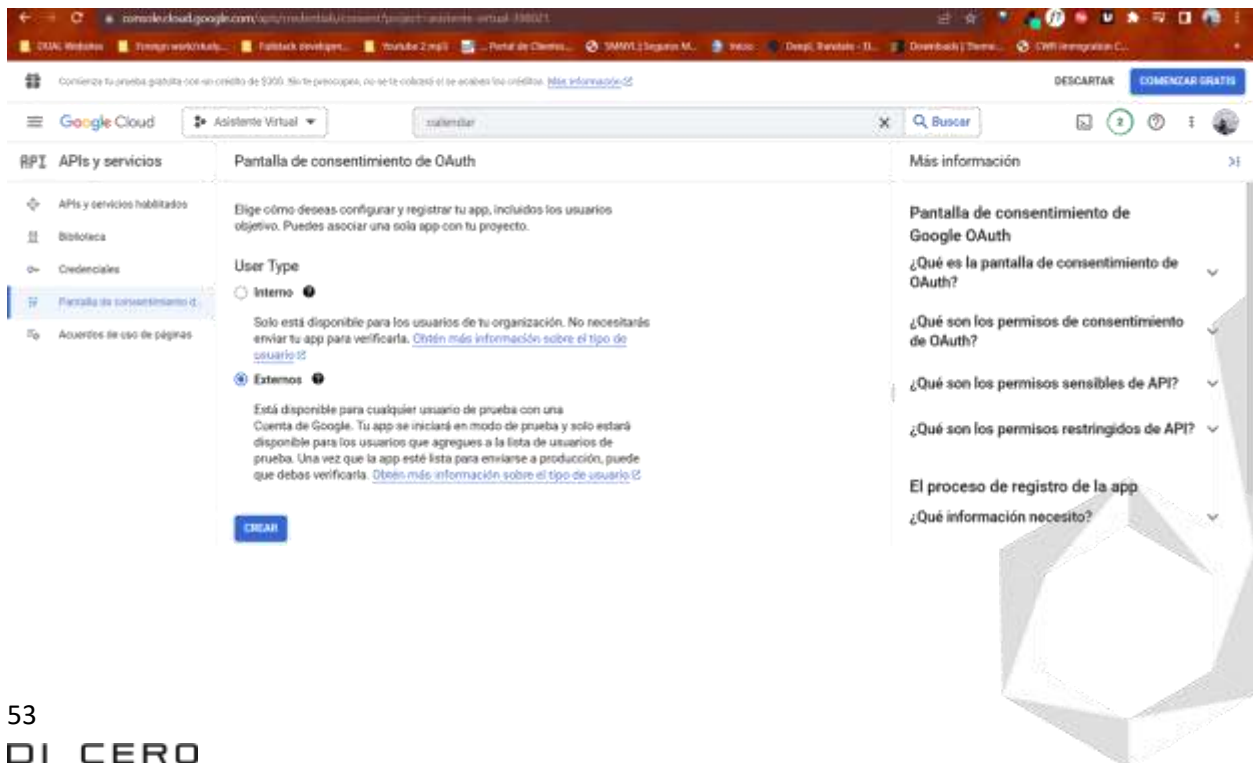
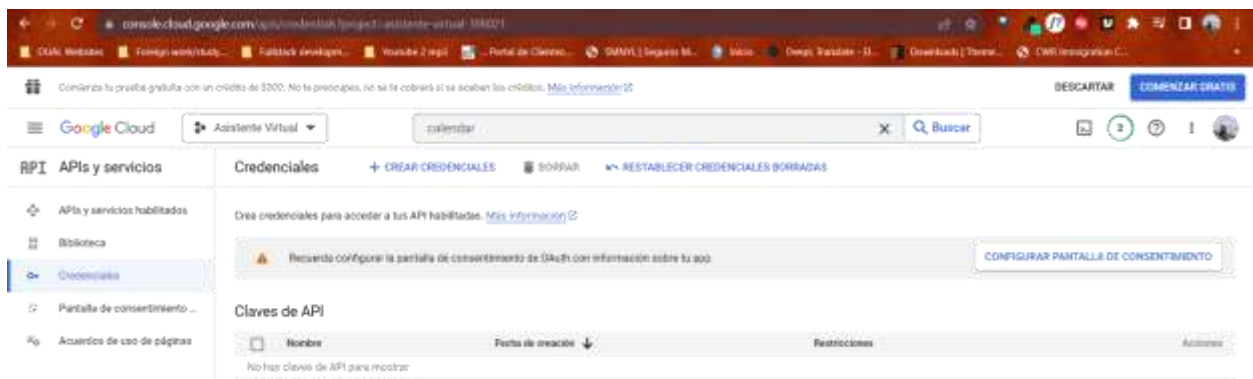


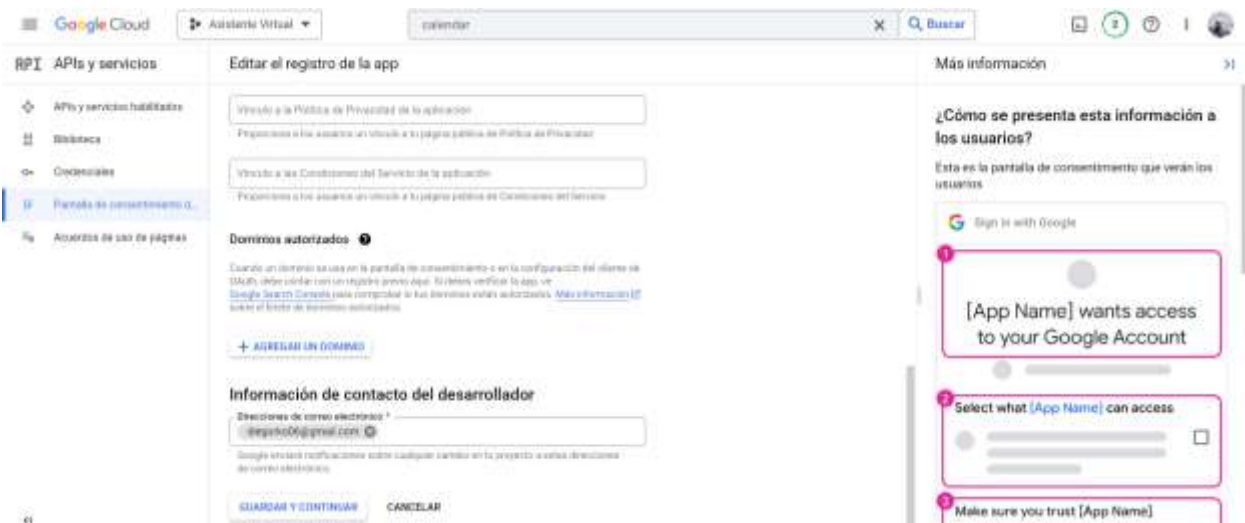
Dentro del nuevo proyecto buscaremos la opción de Google Calendar API y daremos clic sobre ella, para después dar clic en Habilitar:



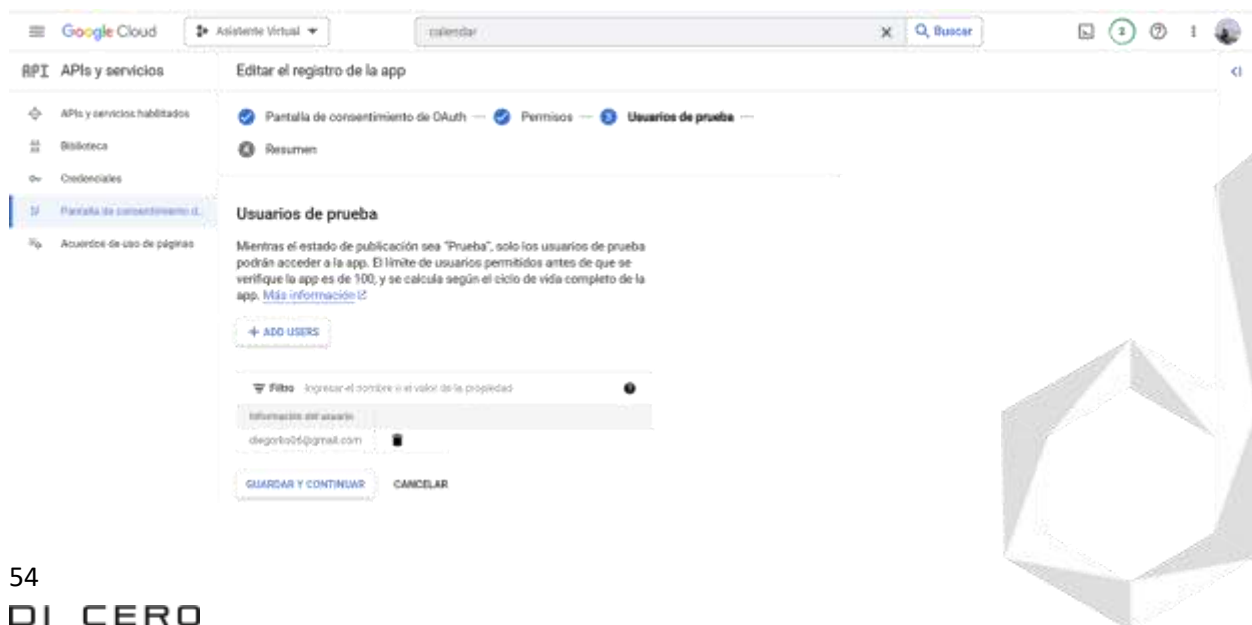
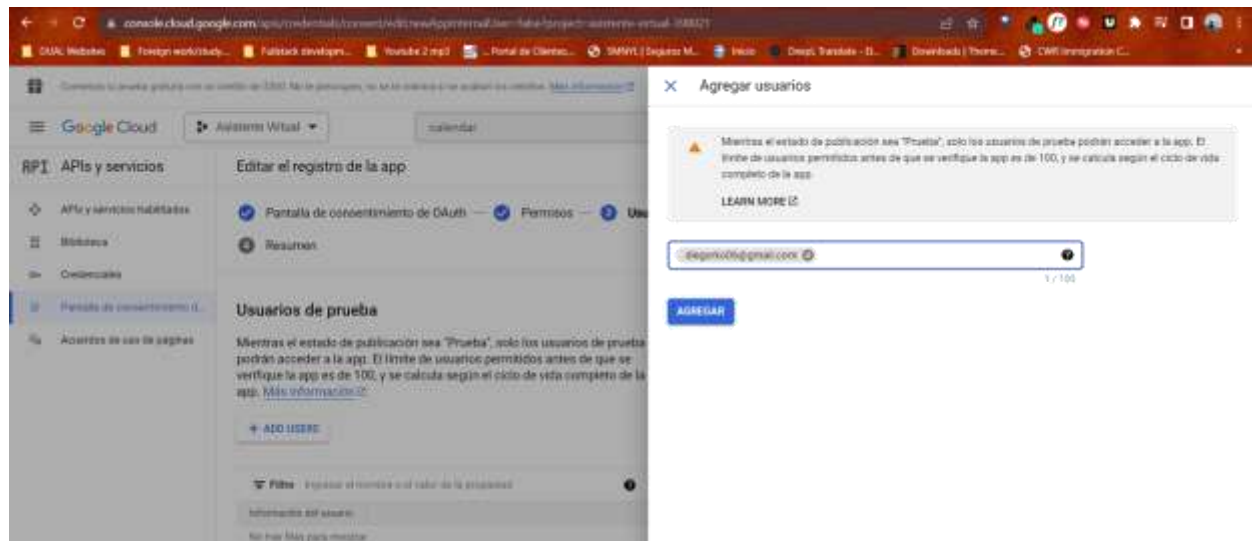


Ahora nos introduciremos a la opción de: Credenciales → Configurar Pantalla de Consentimiento → Pantalla de Consentimiento de OAuth → User Type → Externos → Crear → Información de Contacto del Desarrollador → Ingresaremos nuestro correo de Gmail → Guardar y continuar.

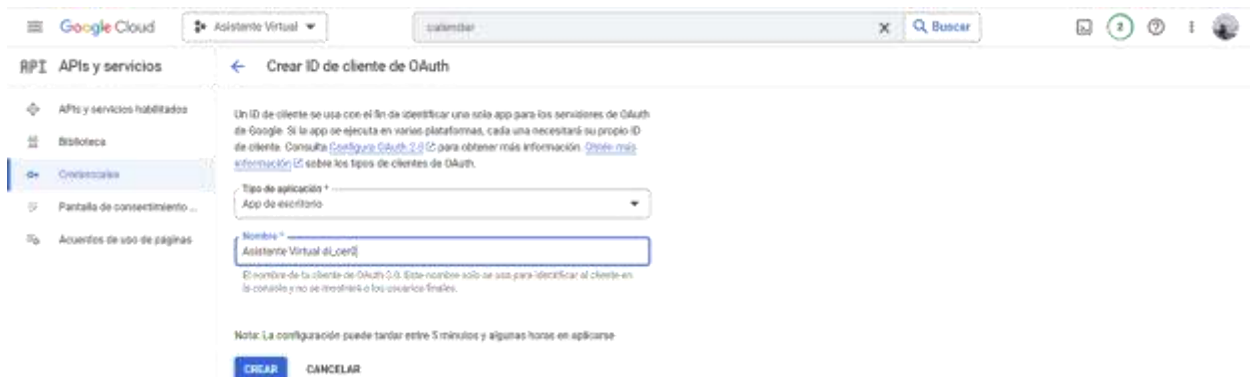




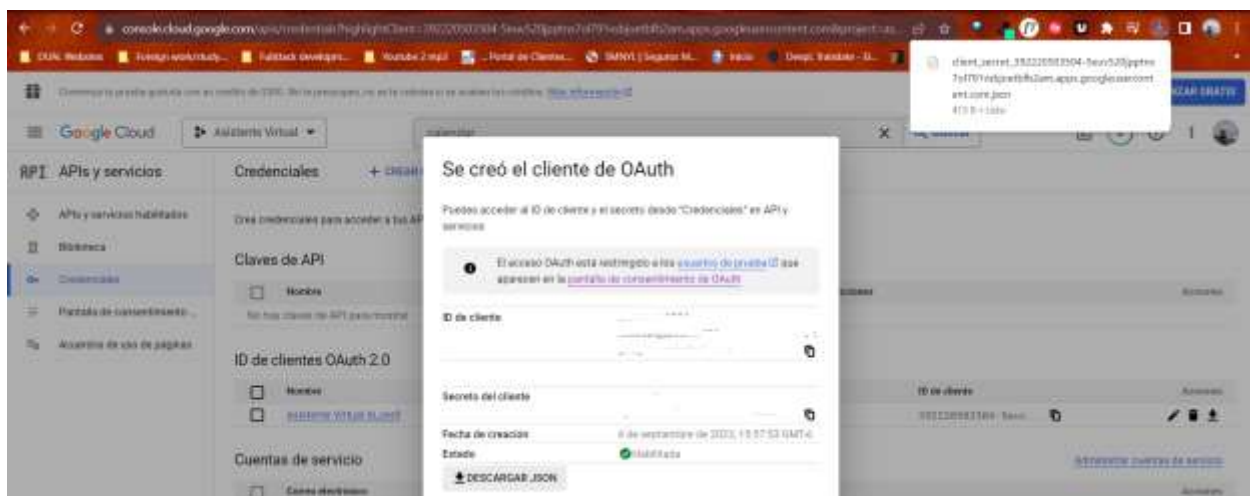
En este punto ya estaremos por finalizar la configuración de la API, para continuar daremos clic en + ADD USERS → Ingresaremos nuestro correo de Gmail → Agregar → Guardar y Continuar.



Finalmente, regresaremos al menú de Credenciales y daremos clic en la opción de: + CREAR CREDENCIALES → ID de cliente de OAuth → Tipo de Aplicación → App de Escritorio → CREAR.



Y para poder utilizar la API creada deberemos descargar el JSON creado y añadirlo a la carpeta de nuestro proyecto.



- **google api python client, auth httpplib y oauthlib - Acción Asistente Virtual - Establecer un Evento en Google Calendar:** La habilidad de establecer un recordatorio para un evento en Google Calendar cuando se le ordene al asistente virtual por medio de la palabra “calendario” se proporciona al ejecutar el siguiente comando en la consola CMD de Windows, que realiza la instalación de las librerías google api client, google http2lib y google oauthlib.

```
pip install --upgrade google-api-python-client google-auth-httpplib2
google-auth-oauthlib
```

```
C:\Users\diego>pip install --upgrade google-api-python-client google-auth-httpplib2 google-auth-oauthlib
Collecting google-api-python-client
  Obtaining dependency information for google-api-python-client from https://files.pythonhosted.org/packages/49/b1/a1364b88787702a308db814742b81fd3b57e4a616c0bbb50a9a3ad97184b/google_api_python_client-2.97.0-py2.py3-none-any.whl.metadata
  Downloading google_api_python_client-2.97.0-py2.py3-none-any.whl.metadata (6.6 kB)
Collecting google-auth-httpplib2
  Downloading google_auth_httplib2-0.1.0-py2.py3-none-any.whl (9.3 kB)
Collecting google-auth-oauthlib
  Downloading google_auth_oauthlib-1.0.0-py2.py3-none-any.whl (18 kB)
Collecting httplib2<1.dev0,>=0.15.0 (from google-api-python-client)
  Downloading httplib2-0.22.0-py3-none-any.whl (96 kB)
Requirement already satisfied: google-auth<3.0.0.dev0,>=1.19.0 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from google-api-python-client) (2.22.0)
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from google-api-python-client) (2.11.1)
Collecting uritemplate<5,>=3.0.1 (from google-api-python-client)
  Downloading uritemplate-4.1.1-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: six in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from google-auth-httpplib2) (1.12.0)
Collecting requests-oauthlib<=0.7.0 (from google-auth-oauthlib)
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: googleapis-common-protos<2.0.dev0,>=1.56.2 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-client) (1.60.0)
Requirement already satisfied: protobuf!=3.20.0,!<3.20.1,!<4.21.0,!<4.21.1,!<4.21.2,!<4.21.3,!<4.21.4,!<4.21.5,<5.0.0.dev0,>=3.19.5 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-client) (4.24.0)
Requirement already satisfied: requests<3.0.0.dev0,>=2.18.0 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests-oauthlib<=0.7.0) (2.28.1)
Requirement already satisfied: oauthlib in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests-oauthlib<=0.7.0) (3.2.0)
Requirement already satisfied: charset-normalizer<3.0.0,>=2.0.0 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests<3.0.0.dev0,>=2.18.0) (2.0.12)
Requirement already satisfied: idna<3.0,>=2.5 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests<3.0.0.dev0,>=2.18.0) (3.4)
Requirement already satisfied: certifi<2021.10.8,>=2021.5.5 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests<3.0.0.dev0,>=2.18.0) (2021.10.8)
Requirement already satisfied: urllib3<1.27.0,>=1.25.9 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests<3.0.0.dev0,>=2.18.0) (1.26.13)
Installing collected packages: httplib2, uritemplate, google-auth-oauthlib, google-auth-httpplib2, google-api-python-client
Successfully installed google-api-python-client-2.97.0 google-auth-httpplib2-0.1.0 google-auth-oauthlib-1.0.0 httplib2-0.22.0 uritemplate-4.1.1
```

Además, con el fin de conocer las zonas horarias pertenecientes a cada localización para asignar la zona horaria a los eventos del calendario se puede consultar el siguiente enlace:

https://en.wikipedia.org/wiki/List_of_tz_database_time_zones



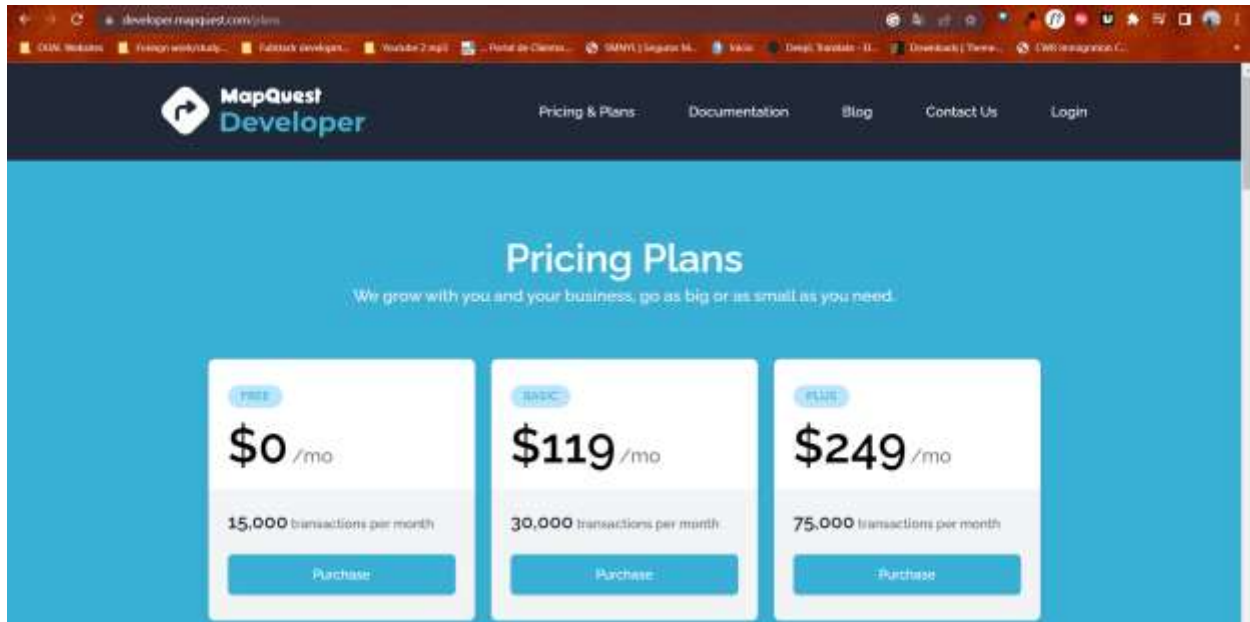
Al terminar de ejecutar el programa por primera vez deberemos dar permiso en nuestra cuenta de Gmail y al finalizar este proceso se mostrará este mensaje en el navegador donde se haya abierto la página web de autenticación:



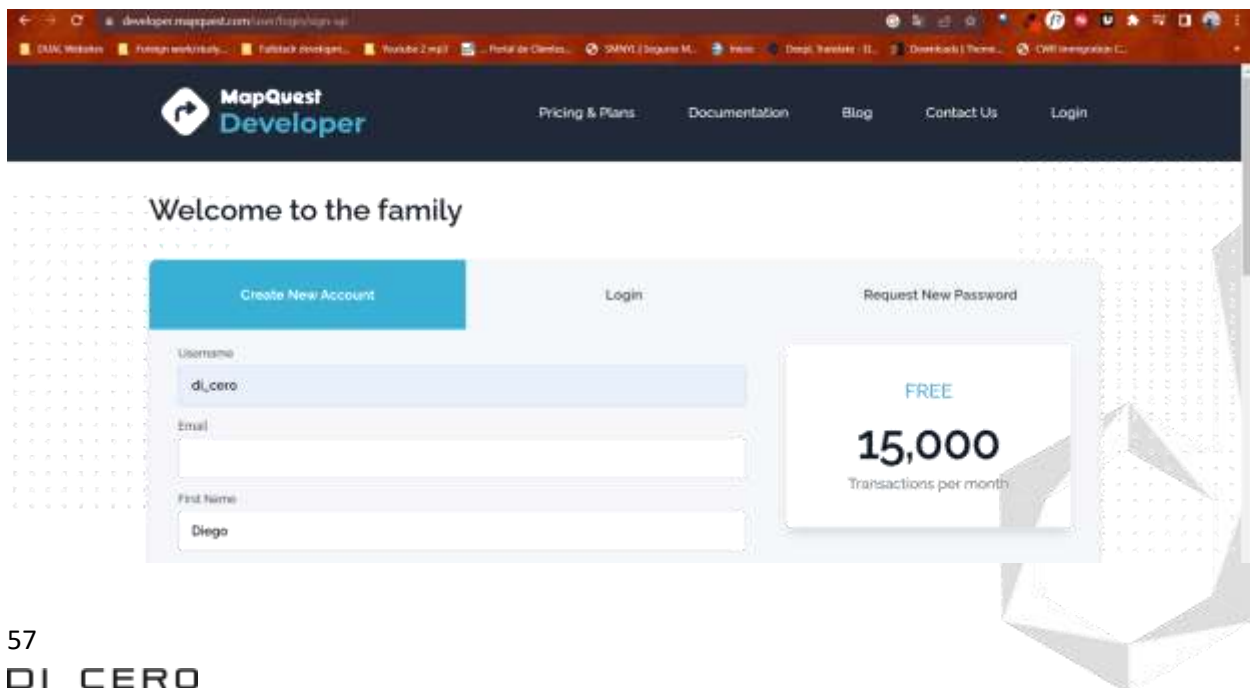
API MapQuest Developer: Herramienta de Google Calendar con Python

Como la API de Google Maps se debe pagar forzosamente, optaremos por utilizar una tecnología alternativa llamada MapQuest Developer, la cual cuenta con un plan gratuito donde se le pueden realizar 15,000 peticiones al mes, si se quiere obtener un mayor número de peticiones se puede contratar otro de sus planes, toda esta información se obtiene a través del siguiente sitio web:

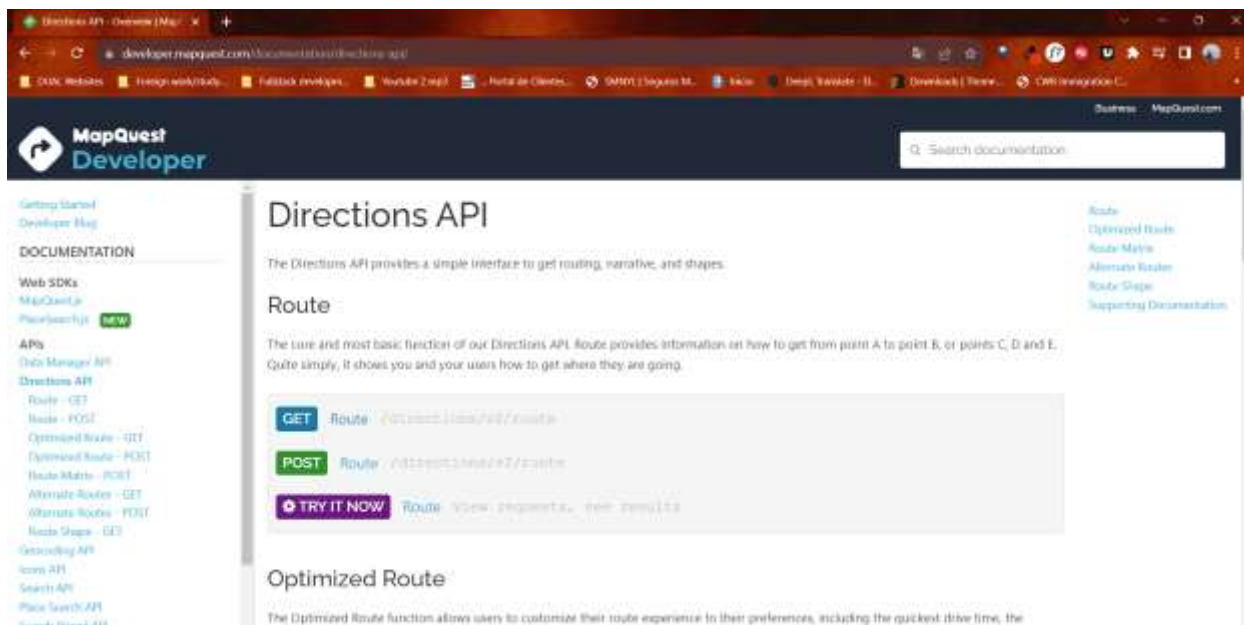
<https://developer.mapquest.com/>



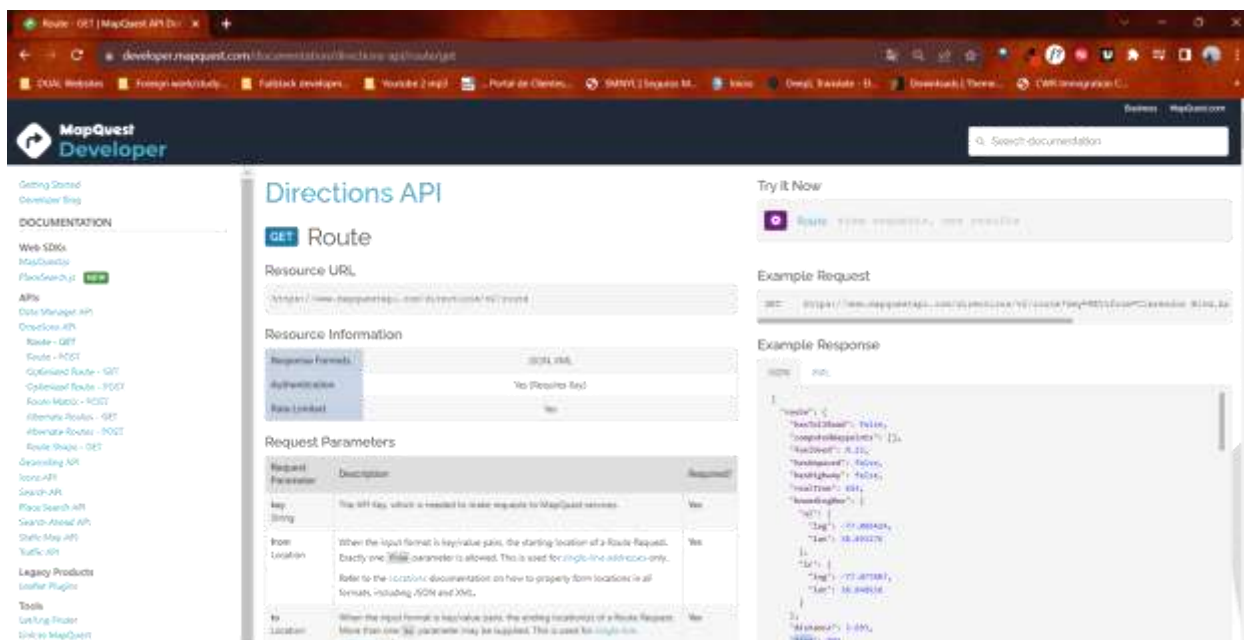
La API REST de MapQuest Developer se comunicará con el programa de Python por medio de una comunicación cliente-servidor usando el Endpoint (URL) de su sitio, a través de los métodos HTTP usuales POST, GET, PUT, DELETE, etc. Pero en este caso el más utilizado será el de GET, que retornará un JSON (JavaScript Object Notation) que incluye los datos de la petición realizada. Para ello primero se debe crear una cuenta.



Una vez creada nuestra cuenta, daremos clic en la opción de Documentation → APIs → Directions API → Route → GET → Route, al realizar peticiones a este endpoint proporcionándole un punto de inicio y uno final nos retornará la distancia entre los dos puntos, el tiempo del recorrido en auto, consumo de combustible, las indicaciones para llegar desde un punto al otro, etc.

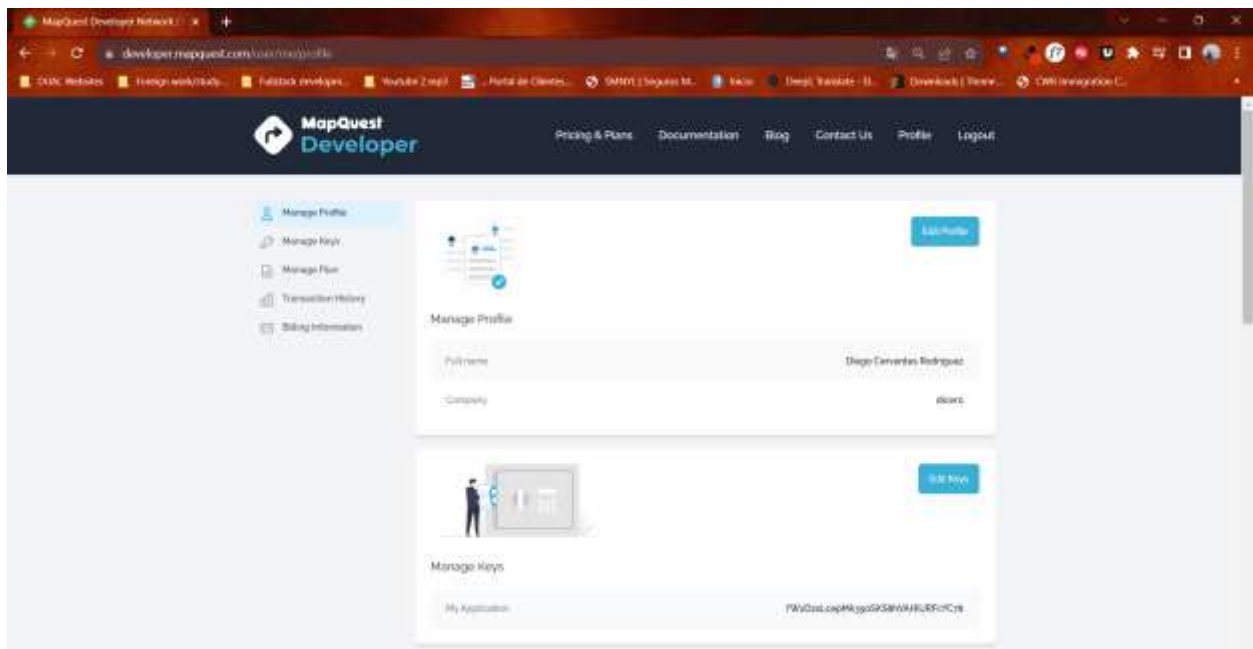


Ya habiendo ingresado a la documentación de este endpoint se nos mostrará su URL, el formato de la respuesta, los parámetros que reciben sus request, ejemplos, etc.

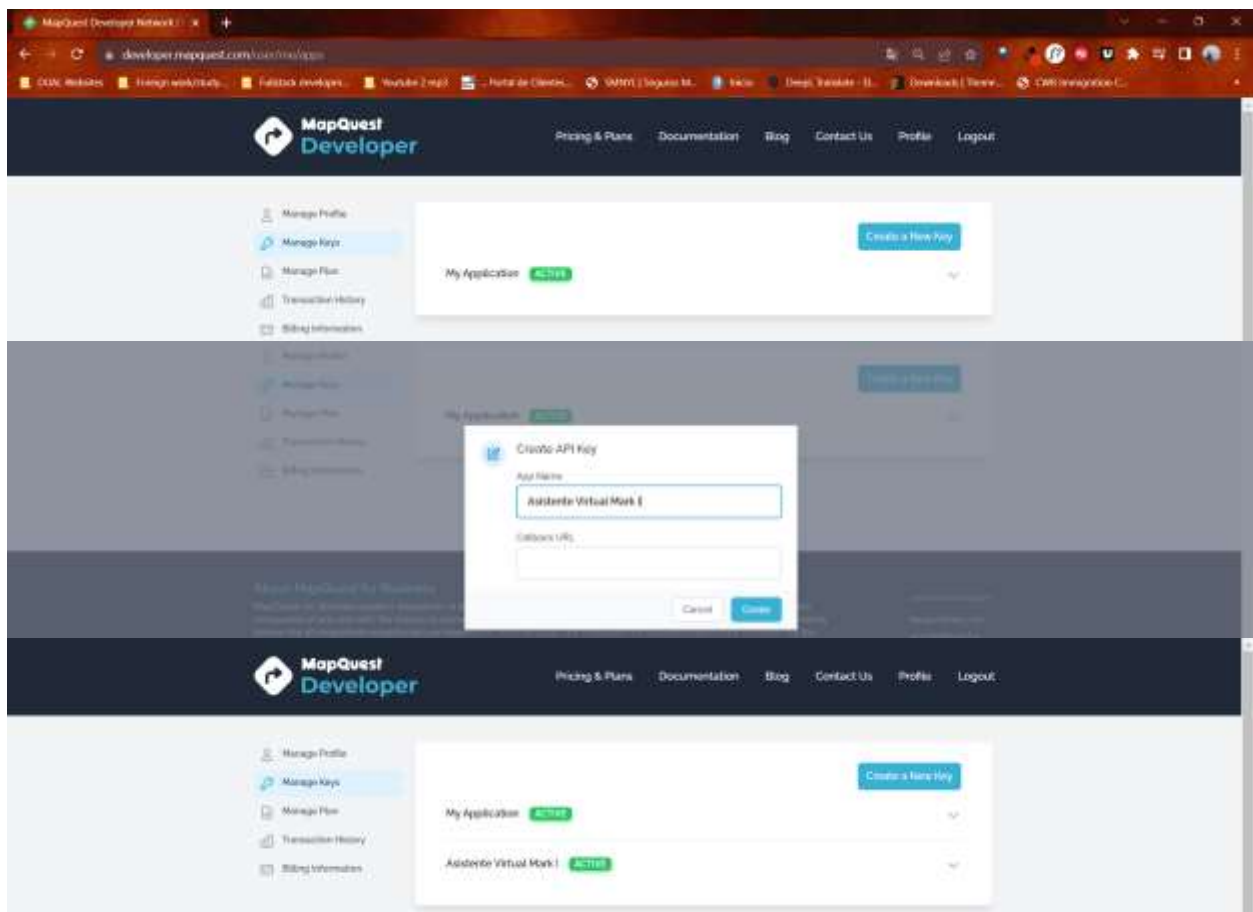


Para poder utilizar esta herramienta debemos crear una API Key, para ello nos introduciremos en la opción de Profile → Manage Keys → Edit Keys.



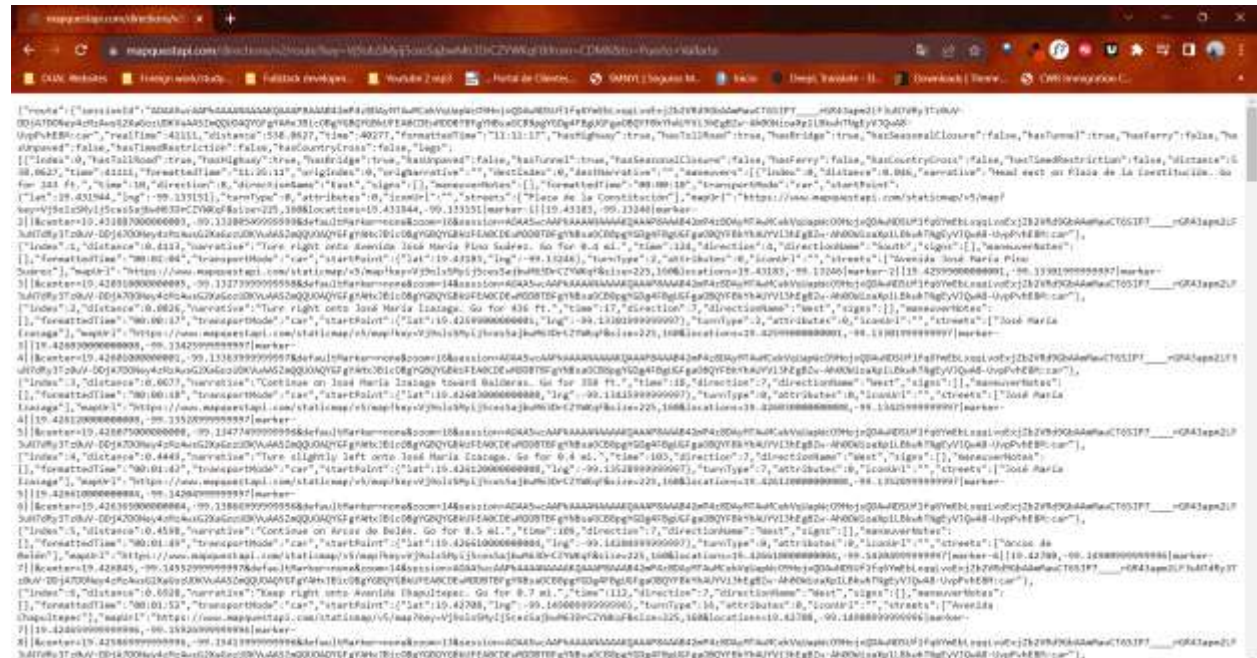


Luego daremos clic en la opción de Create a New Key → App Name (Nombre del proyecto) → Callback URL (se deja vacía) → Create y al desplegar la pestaña que tiene el nombre de la app que acabamos de crear podremos extraer su API key donde dice Costumer Key.



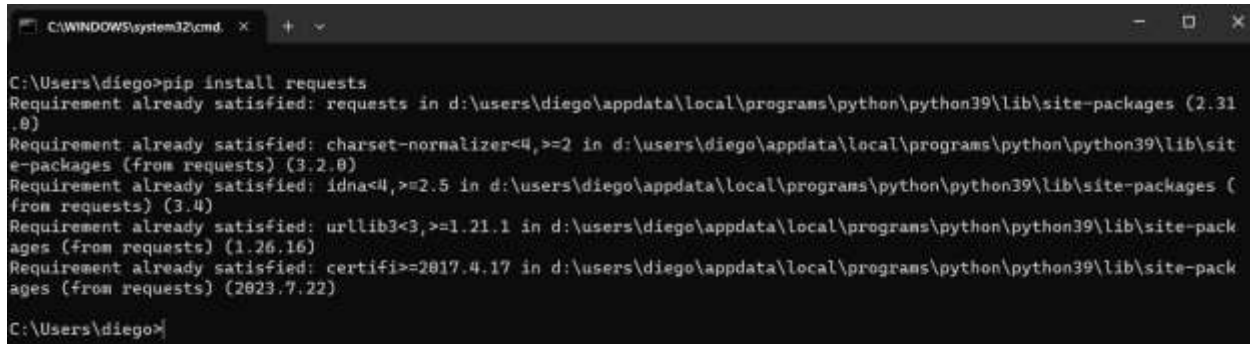
Finalmente, para poder observar de una forma más amigable los datos retornados por la API, podemos instalar la extensión de JSON Viewer en el navegador de Chrome:

<https://chrome.google.com/webstore/category/extensions>



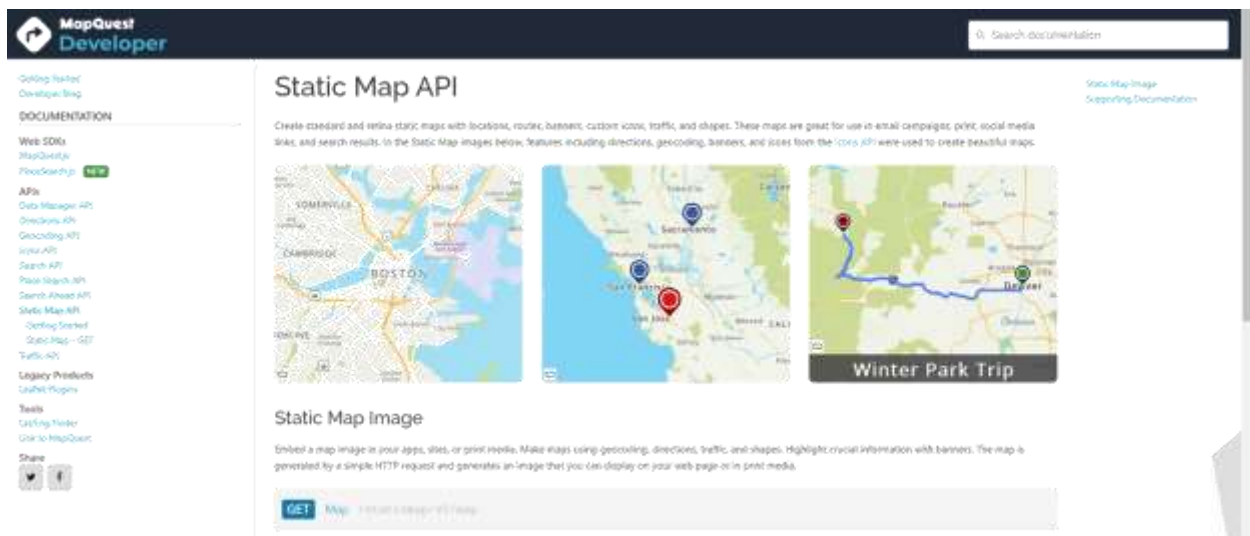
- **requests - Acción Asistente Virtual - Realizar peticiones a la API de MapQuest:** La habilidad de obtener la distancia entre los dos puntos, el tiempo del recorrido en auto, consumo de combustible, las indicaciones para llegar desde un punto al otro, etc. cuando se le ordene al asistente virtual por medio de la palabra “mapa” se proporciona al ejecutar el siguiente comando en la consola CMD de Windows, que realiza la instalación de la librería request que permite hacer consultas a una API a través de una URL (endpoint). También se utiliza la librería urllib, pero esa no se debe instalar, ya viene incluida con Python.

```
pip install requests
```



```
C:\Users\diego>pip install requests
Requirement already satisfied: requests in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests) (1.26.16)
Requirement already satisfied: certifi<=2017.4.17 in d:\users\diego\appdata\local\programs\python\python39\lib\site-packages (from requests) (2023.7.22)
C:\Users\diego>
```

Con las instrucciones anteriores obtenemos las indicaciones y datos para llegar de un punto a otro, pero para obtener un mapa que muestre la ruta completa o en cachos debemos ingresar a la opción de Documentation → APIs → Static Map API → GET → Map → Resource URL → Adding a Route to the Map. En esa parte es donde se mostrarán los pasos que se deben seguir para obtener una URL que muestre una imagen estática con la ruta que se deben seguir.



- **Pillow - Acción Asistente Virtual - Mostrar la imagen del Mapa Retornado por la API de MapQuest:** La habilidad de mostrar una imagen del mapa que muestra el recorrido cuando se le ordene al asistente virtual por medio de la palabra “mapa” se proporciona al ejecutar el siguiente comando en la consola CMD de Windows, que realiza la instalación de la librería pillow que permite mostrar una imagen a través de una URL (endpoint). También se utiliza la librería io pero esa no se debe instalar, ya viene incluida con Python.

```
pip install Pillow
```

```
C:\WINDOWS\system32\cmd. X + v
C:\Users\diego>pip install pillow
Collecting pillow
  Obtaining dependency information for pillow from https://files.pythonhosted.org/packages/8f/b8/1bf1012eee3059d150194d1fab148f553f3df42cf412e4e6656c772afad9/Pillow-10.0.0-cp39-cp39-win_amd64.whl.metadata
  Using cached Pillow-10.0.0-cp39-cp39-win_amd64.whl.metadata (9.6 kB)
  Using cached Pillow-10.0.0-cp39-cp39-win_amd64.whl (2.5 MB)
Installing collected packages: pillow
Successfully installed pillow-10.0.0
C:\Users\diego>
```

OpenCV: Visión Artificial con Python

Para realizar todas las funciones de visión artificial del asistente virtual se utiliza la herramienta de OpenCV, específicamente las habilidades que se le dará al asistente virtual será reconocimiento de colores y reconocimiento facial a través de su entrenamiento utilizando técnicas de machine learning.

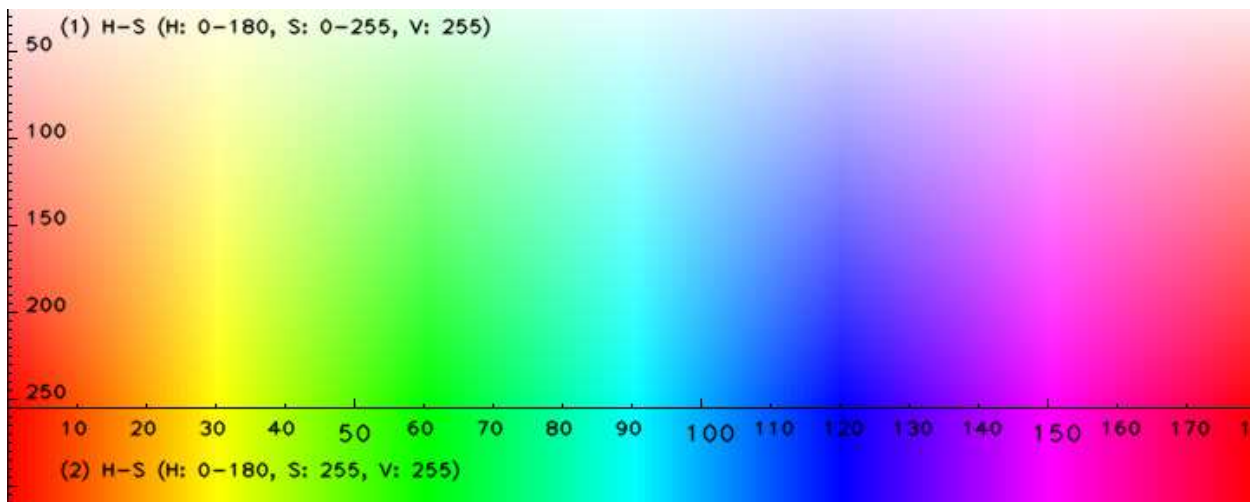
- **OpenCV - Acción Asistente Virtual - Visión artificial:** La habilidad de activar la cámara o analizar una imagen para obtener algún dato de ella se realiza a través de la librería opencv, para ello se debe ejecutar el siguiente comando en la consola CMD de Windows.

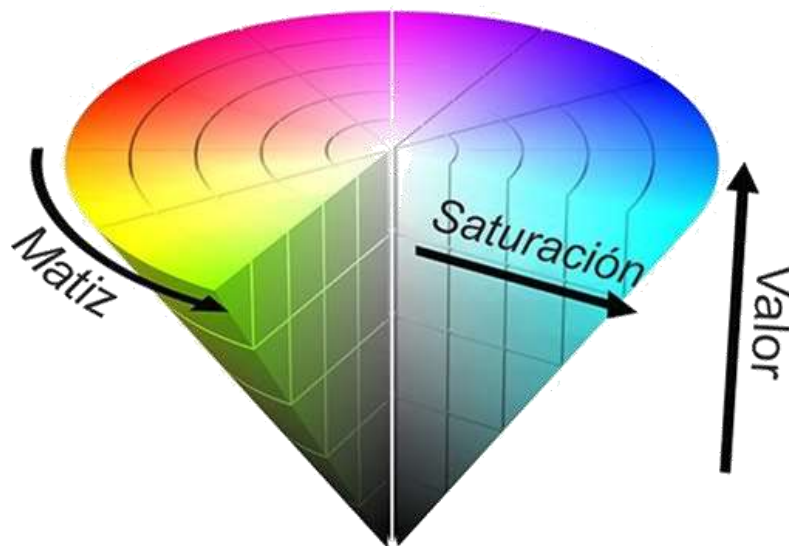
```
pip install opencv-python
```

Para realizar detectar los colores en una imagen o video la mejor manera de hacerlo es a través del código de color HSV (Hue, Saturation y View), ya que este no solo considera el color sino también su intensidad, luz, etc.

- **Hue = Matiz:** Representa todos los tonos de colores y abarca valores de 0 a 180.
- **Saturation = Saturación:** Representa la intensidad del color y abarca valores de 0 a 255.
- **View = Valor:** Representa la luz y abarca valores también de 0 a 255.

En la siguiente gráfica el **Matiz (Hue)** corresponde al eje x y la **Saturación (Saturation)** al eje y va de arriba hacia abajo, la luz igual se varía, pero eso no se ve en la gráfica 2D.





- **imutils - Acción Asistente Virtual - Reconocimiento facial:** La habilidad de reconocer un rostro cuando se le ordene al asistente virtual por medio de la palabra “reconocimiento” se proporciona al ejecutar el siguiente comando en la consola CMD de Windows, que realiza la instalación de la librería imutils que permite acelerar el procesamiento de imágenes.

```
pip install imutils
```

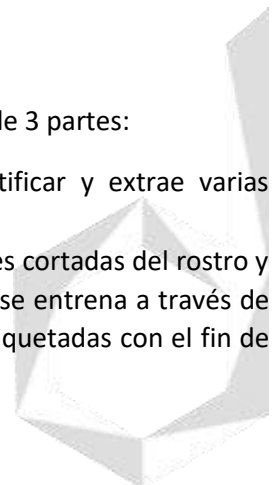
```
C:\WINDOWS\system32\cmd x + ~
C:\Users\diego>pip install imutils
Collecting imutils
  Downloading imutils-0.5.4.tar.gz (17 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: imutils
  Building wheel for imutils (setup.py) ... done
  Created wheel for imutils: filename=imutils-0.5.4-py3-none-any.whl size=25848 sha256=b4e9c996d9dc59682284459fa02842e85
  20a47ddb223df7c4842dd30a9197151
  Stored in directory: c:\users\diego\appdata\local\pip\cache\wheels\4b\45\2d\4a078a801d3a3d93f033d3ee9728f470f514826e89
  952df3ea
Successfully built imutils
Installing collected packages: imutils
Successfully installed imutils-0.5.4
C:\Users\diego>
```

Para el reconocimiento facial se utilizará un modelo llamado Haarcascade, el cual se basa en archivos XML donde se ha guardado la información necesaria para realizar reconocimiento facial en situaciones específicas, como reconocimiento de rostros de manera frontal, lateral, de ojos, boca, etc. El cual se encuentra localizado en el siguiente perfil de GitHub:

<https://github.com/opencv/opencv/tree/4.x/data/haarcascades>

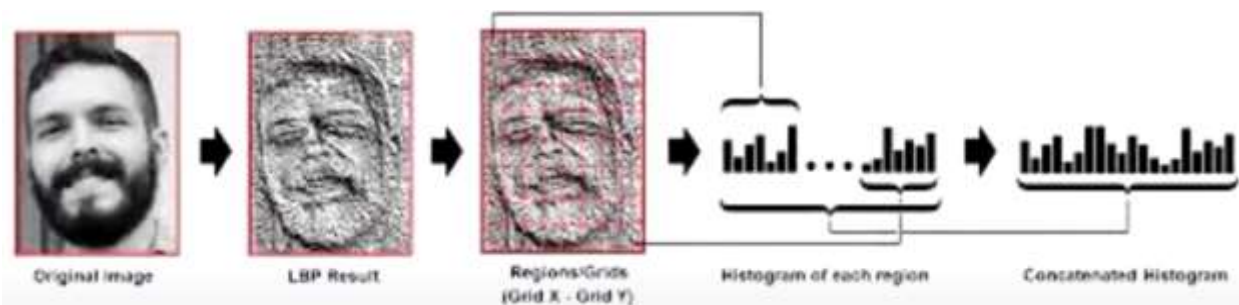
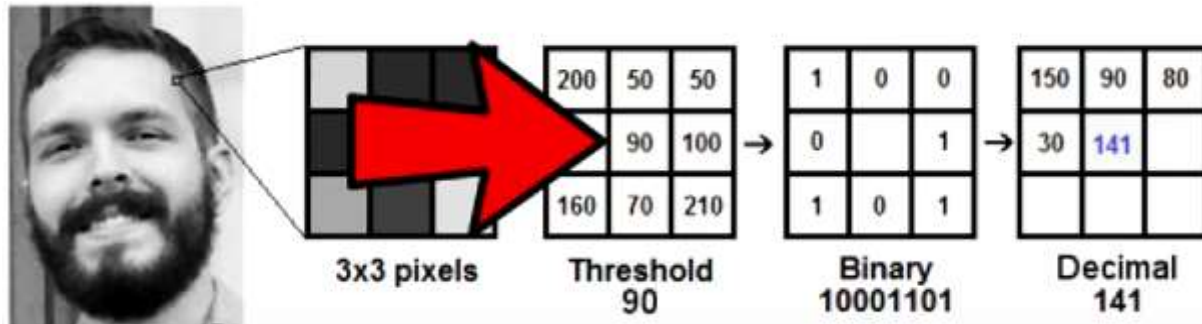
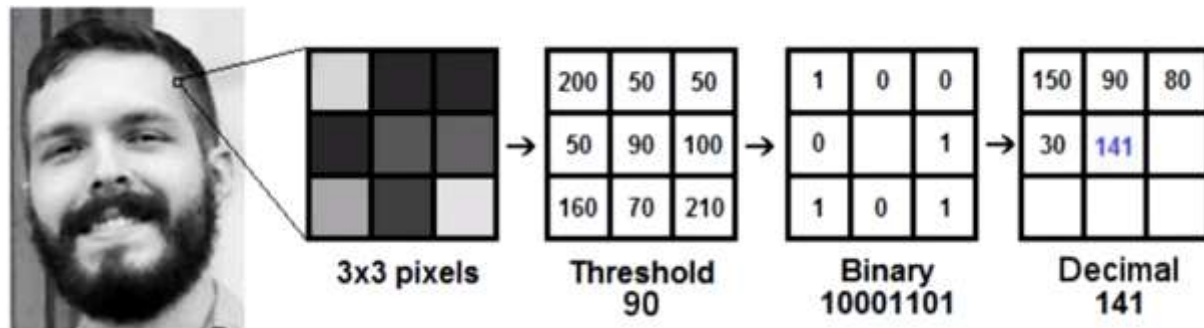
El entrenamiento de machine learning para el reconocimiento facial se compone de 3 partes:

- La primera se alimenta de un video del usuario que se quiere identificar y extrae varias imágenes de su rostro.
- Posteriormente se crea un archivo Python que analice todas esas imágenes cortadas del rostro y las convierta a un objeto LBPH (Local Binary Patterns Histogram), el cual se entrena a través de todas las imágenes tomadas previamente, para ello todas deben estar etiquetadas con el fin de



diferenciarse una de la otra, luego procesa toda esa información y obtiene una representación matemática en forma de histograma que represente cada rostro. La forma en la que el objeto LBPH realiza la conversión es la siguiente:

- Primero toma las 3 capas de la matriz RGB de la imagen original y la convierte a su escala de grises para que en vez de que se analice una matriz tridimensional se tenga una sola, cuyos valores podrán ir de 0 a 255 para considerar cada matriz de gris.
- Luego esos valores de 0 a 255 de la capa de grises, los transformará a un equivalente binario dependiendo de si es mayor o menor al píxel central de la vecindad de la imagen, considerándolo así como su umbral (Threshold) de binarización.
- Finalmente, ese número binario se cortará y se transformará en un número decimal, para así obtener el histograma de la imagen, basado en los contornos del rostro.



- Finalmente se creará un programa de Python que a través del modelo pre entrenado analice las imágenes del video captado en tiempo real (llamado frame) y analice si el rostro que está captando a través de nuevamente el modelo Haarcascade de OpenCV es reconocido.

Asistente Virtual - Mark I: Google Calendar, Mapas, Visión Artificial y Machine Learning

```
# -*- coding: utf-8 -*-

#En Python se introducen comentarios de una sola linea con el simbolo #.
#La primera línea de código incluida en este programa se conoce como declaración de codificación o codificación
#de caracteres. Al especificar utf-8 (caracteres Unicode) como la codificación, nos aseguramos de que el archivo
#pueda contener caracteres especiales, letras acentuadas y otros caracteres no ASCII sin problemas, garantizando
#que Python interprete correctamente esos caracteres y evite posibles errores de codificación.
#Se puede detener una ejecución con el comando [CTRL] + C puesto en consola, con el comando "cls" se borra su
#historial y en Visual Studio Code con el botón superior derecho de Play se corre el programa.
#Para comentar en Visual Studio Code varias líneas de código se debe pulsar:
#[CTRL] + K (VSCode queda a la espera). Después pulsa [CTRL] + C para comentar y [CTRL] + U para descomentar.

#IMPORTACIÓN DE LIBRERÍAS:
#REPRODUCIR VIDEO EN YOUTUBE:
#pywhatkit: Pywhatkit es una biblioteca de Python que permite enviar mensajes de texto a través de WhatsApp
#incluso si no estás en su lista de contactos, realizar búsquedas en la web, reproducir canciones en YouTube,
#realizar búsquedas en Wikipedia, ejecutar comandos en consola, etc.
import pywhatkit

#threading: Librería que permite crear y gestionar los hilos (threads) de un programa. Los hilos son subprocesos
#que le permiten a un programa realizar múltiples tareas al mismo tiempo, esto se añade debido a la clase de
#visión artificial, ya que si esto no se añade, el asistente virtual detendrá su ejecución mientras se ejecuta
#la clase visionArtificialAsistente.
import threading

#IMPORTACIÓN DE CLASES: Cuando se quiera importar una clase, el nombre de esta no puede empezar con un número,
#sino cuando la quiera importar obtendré un error y se va accediendo a las carpetas o también llamados paquetes
#en la programación orientada a objetos (POO), por medio de puntos:
# - Directorio normal:      carpeta1/carpeta2/carpeta3
# - Directorio paquetes:   carpeta1.carpeta2.carpeta3
#La parte del directorio se coloca después de la palabra reservada from y la clase a importar después de import.
from POO_AsistenteVirtual_MarkI.oidoAsistente import EscucharMicrofono
from POO_AsistenteVirtual_MarkI.vozAsistente import vozWindows
from POO_AsistenteVirtual_MarkI.cerebroLangchainAsistente import cerebro_OpenAI
from POO_AsistenteVirtual_MarkI.alarmaAsistente import widgetAlarma
from POO_AsistenteVirtual_MarkI.notasAsistente import widgetEscribirNotas
from POO_AsistenteVirtual_MarkI.whatsappAsistente import widgetWhatsApp
from POO_AsistenteVirtual_MarkI.mapaAsistente import widgetDireccionesMapa
from POO_AsistenteVirtual_MarkI.googleCalendarAsistente import widgetGoogleCalendar
from POO_AsistenteVirtual_MarkI.visionArtificialAsistente import visionArtificial
from POO_AsistenteVirtual_MarkI.reconocimientoFacialAsistente import machineLearning

#INYECCIÓN DE DEPENDENCIAS:
#Inyección de dependencias: Se refiere a un patrón de diseño donde, en lugar de que una clase cree directamente
#sus dependencias, osea los objetos o variables de las que depende para funcionar; estas se le suministren desde
```



```

#el exterior. Esto se hace típicamente a través de parámetros de constructor, métodos o propiedades.
#En este caso la inyección de dependencias se utiliza para activar la función de crear eventos en Google Calendar,
#ya que la clase googleCalendarAsistente que se encuentra en el paquete POO_AsistenteVirtualMarkI necesita
#forzosamente de un método perteneciente a una clase distinta llamada googleCalendar_auth, que además se encuentra
#en otro paquete, por ello es que en este archivo se declarará un objeto de dicha clase y este será recibido como
#parámetro de la clase widgetGoogleCalendar para que la función se pueda activar.
import API_Keys.googleCalendar_auth
autenticacionGoogleCalendar = API_Keys.googleCalendar_auth.servicioAPIGoogleCalendar()

#IMPORTACIÓN DE DATOS SENSIBLES:
#IMPORTACIÓN DE LLAVE: Cuando se quiera utilizar una API que utiliza un key, por seguridad es de buenas
#prácticas declararla en un archivo externo, además cabe mencionar que el nombre de dicho archivo y constante no
#pueden empezar con un número, sino cuando la quiera importar obtendré un error y se va accediendo a sus carpetas
#por medio de puntos:
# - Directorio normal:      carpeta1/carpeta2/carpeta3
# - Directorio paquetes:    carpeta1.carpeta2.carpeta3
#La parte del directorio se coloca después de la palabra reservada import y posteriormente se manda a llamar sus
#variables o constantes de igual manera a través de un punto.
import API_Keys.Llaves_ChatGPT_Bard_etc
#ChatGPT API key, Teléfono di_cer0 y MapQuest API key.
ApiKey = API_Keys.Llaves_ChatGPT_Bard_etc.LlaveChatGPT
contacto = API_Keys.Llaves_ChatGPT_Bard_etc.telefonoDicer0
ApiKeyMapQuest = API_Keys.Llaves_ChatGPT_Bard_etc.LlaveMapQuest

#__name__ == __main__: Método main, esta función es super importante ya que sirve para instanciar las clases del
#programa y ejecutar sus métodos, en python pueden existir varios métodos main en un solo programa, aunque no es
#una buena práctica.
if (__name__ == "__main__"):
    #OBJETOS DE LAS FUNCIONES DEL ASISTENTE VIRTUAL:
    oidoAsistenteVirtual = EscucharMicrofono() #Instancia de la clase propia EscucharMicrofono.
    vozAsistenteVirtual = vozWindows() #Instancia de la clase propia vozWindows.
    cerebroAsistenteVirtual = cerebro_OpenAI(ApiKey) #Instancia de la clase propia cerebro_OpenAI.
    alarmaAsistenteVirtual = widgetAlarma() #Instancia de la clase propia widgetAlarma.
    horaAlarma = None #Variable que guarda el estado de la alarma.
    notasAsistenteVirtual = widgetEscribirNotas() #Instancia de la clase propia widgetAlarma.
    whatsappAsistenteVirtual = widgetWhatsApp(contacto) #Instancia de la clase propia widgetWhatsApp.
    mapaAsistenteVirtual = widgetDireccionesMapa(ApiKeyMapQuest) #Instancia de la clase widgetDireccionesMapa.
    #Instancia de la clase widgetGoogleCalendar con inyección de dependencias de la clase googleCalendar_auth.
    calendarioAsistenteVirtual = widgetGoogleCalendar(autenticacionGoogleCalendar)
    visionColoresAsistenteVirtual = visionArtificial() #Instancia de la clase visionArtificial.
    visionRostrosAsistenteVirtual = machineLearning() #Instancia de la clase machineLearning.
    vozAsistenteVirtual.hablar("Quiubolas di0, soy TIMI Mark 1, tu asistente virtual, puedo responder tus " +
                                "preguntas con ChatGPT, reproducir canciones o videos en YouTube, programar " +
                                "alarmas, escribir en mi archivo de notas, mandar mensajes por WhatsApp, mostrar "+
                                "imagenes con mapas de rutas, crear recordatorios en Google Calendar y activar " +

```




```

        "mi visión artificial para reconocer colores o reconocimiento facial de tu " +
        "carita chula..." +
        "En que te puedo ayudar?")

while True:
    try:
        #EscucharMicrofono.oidoAsistenteVirtual(): Método propio de la clase EscucharMicrofono que se encarga
        #de escuchar lo que dice el usuario, almacenar eso en un archivo temporal y luego pasárselo al modelo
        #Whisper para que lo transcriba a texto.
        preguntaUsuario = oidoAsistenteVirtual.oidoAsistenteVirtual()
        print(str(preguntaUsuario) + "\n\n")

        #Si se reconoce que el usuario dice bye Timmy, se rompe el bucle while y se termina la ejecución del
        #programa.

        #any():Función que recibe un bucle for que recorrerá un diccionario, lista o tupla para después
        #retornar un valor booleano True cuando al menos uno de sus elementos sea True o distinto a None.
        if any(palabra in preguntaUsuario for palabra in ["adiós, timmy.", "bye, timmy.", "adiós, tími.", "adiós, dimi.",
"adiós, teamy.", "¡bye, timmy!", "adios timmy", "bye timmy"]):
            vozAsistenteVirtual.hablar("Bye di0, un gusto haberte ayudado...")
            print("-----Bye di_cer0, un gusto haberte ayudado...-----")
            break

        #Si se reconoce que el usuario dice la palabra reproduce, el asistente virtual por medio de la
        #librería pywhatkit reproduce el video que el usuario dijo en YouTube.
        elif (("reproduce") in preguntaUsuario):
            #.replace().lower(): Lo que hace el método replace() es reemplazar todas las palabras que
            #aparezcan en un string en otra cadena específica y el método lower() sirve para convertir todas
            #las letras de una cadena en minúsculas, esto es importante hacerlo porque al reproducir una
            #canción o mandar una búsqueda a internet, siempre es mejor tener puras minúsculas.
            cancionYoutube = preguntaUsuario.replace('reproduce', '').lower()
            vozAsistenteVirtual.hablar("Ok di0, voy a reproducir el video que pediste en YouTube")
            #pywhatkit.playonyt(): El método playonyt() de la librería pywhatkit permite reproducir un
            #video de YouTube en el navegador web predeterminado del sistema.
            pywhatkit.playonyt(cancionYoutube)
            print("-----Ok di_cer0, ya reproducí la canción del artista que pediste en YouTube...-----")

        #Si se reconoce que el usuario dice la palabra alarma, el asistente virtual por medio de una
        #de la clase widgetAlarma programa una alarma en la hora que el usuario haya indicado.
        elif (("alarma") in preguntaUsuario):
            #widgetAlarma.programarAlarma(): Método de la clase propia widgetAlarma que ejecuta una alarma
            #que solamente puede ser detenida al presionar la tecla de SpaceBar.
            horaAlarma = alarmaAsistenteVirtual.programarAlarma(preguntaUsuario)
            vozAsistenteVirtual.hablar("Hola di0, alarma puesta a las " + horaAlarma)
            print("-----Ok di_cer0, ya he establecido la alarma que me dijiste...-----")

        #Si se reconoce que el usuario dice la palabra ruta, llegar, llego, direcciones, distancia, viaje,
        #mapa, tráfico, navegación, entre otras. el asistente virtual por medio de una de la clase

```



```

#widgetAlarma programa una alarma en la hora que el usuario haya indicado.

#any():Función que recibe un bucle for que recorrerá un diccionario, lista o tupla para después
#retornar un valor booleano True cuando al menos uno de sus elementos sea True o distinto a None.
elif any(palabra in preguntaUsuario for palabra in ["llegó", "llego", "llegar", "ruta", "direcciones", "distancia",
"viaje", "mapa", "tráfico", "navegación"]):
    vozAsistenteVirtual.hablar("Dame dos, deja proceso los datos de la ruta...")
    instruccionesMapa = mapaAsistenteVirtual.direccionesMapa(preguntaUsuario)
    vozAsistenteVirtual.hablar("Hola di0, " + instruccionesMapa)
    print("-----Ok di_cer0, ya te he dado las instrucciones de la ruta que me pediste.....")

#Si se reconoce que el usuario dice la palabra agenda, calendario, calendar, recordatorio, entre
#otras. el asistente virtual por medio de una de la clase widgetGoogleCalendar programa un evento en
#Google Calendar a través de su API con el título, descripción, fecha y hora de inicio y finalización
#que el usuario haya indicado.
#any():Función que recibe un bucle for que recorrerá un diccionario, lista o tupla para después
#retornar un valor booleano True cuando al menos uno de sus elementos sea True o distinto a None.
elif any(palabra in preguntaUsuario for palabra in ["crea", "agenda", "calendario", "calendar", "recordatorio"]):
    resultadoCalendario = calendarioAsistenteVirtual.crearEvento(preguntaUsuario)
    vozAsistenteVirtual.hablar("Hola di0, " + resultadoCalendario)
    print("-----Ok di_cer0, ya he creado el recordatorio que me pediste.....")

#Si se reconoce que el usuario dice la palabra visión, visión artificial, ver color, etc. el asistente
#virtual por medio de una de la clase machineLearning enciende la cámara del ordenador para observar
#el rostro del usuario, si lo reconoce pondrá su nombre en pantalla, pero si no lo reconoce hará sonar
#una alarma.
elif any(palabra in preguntaUsuario for palabra in ["vision", "visión", "visión artificial", "vision artificial",
"ver color", "ver colores"]):
    #visionArtificial.reconocerColoresVideo(): Método de la clase propia visionArtificial que activa
    #la cámara del ordenador para reconocer los colores de las figuras que se muestran en el video.
    #Para no se detenga la ejecución del asistente virtual se hace uso de la librería threading.
    #threading.Thread(): El constructor de la clase Thread se utiliza para crear un nuevo hilo en el
    #programa, que se ejecutará en paralelo con el hilo principal del programa u otros hilos. Para ello
    #en el constructor se debe declarar una función que se ejecute en el nuevo hilo, sin poner sus
    #paréntesis.
    hiloVision = threading.Thread(target = visionColoresAsistenteVirtual.reconocerColoresVideo)
    vozAsistenteVirtual.hablar("Hola di0, he activado mi visión artificial para reconocer colores, para cerrar
la cámara da clic en la tecla c.")
    #threading.Thread().start(): Una vez creado el nuevo hilo con el constructor de la clase Thread, se
    #inicializará su ejecución con el método start().
    hiloVision.start()
    print("-----Ok di_cer0, he activado mi visión artificial para reconocer colores.....")

#Si se reconoce que el usuario dice la palabra facial, reconocimiento, rostro, cara, etc. el asistente
#virtual por medio de una de la clase visionArtificial enciende la cámara del ordenador para observar
#los colores rojo, verde, azul o amarillo de su entorno.

```



```

elif any(palabra in preguntaUsuario for palabra in ["facial", "reconocimiento", "reconocimiento facial", "rostro",
"cara"]):

    #machineLearning.reconocimientoFacial(): Método de la clase propia machineLearning que activa
    #la cámara del ordenador para reconocer los rostros de los usuarios que se muestren en el video.
    #Para no se detenga la ejecución del asistente virtual se hace uso de la librería threading.
    #threading.Thread(): El constructor de la clase Thread se utiliza para crear un nuevo hilo en el
    #programa, que se ejecutará en paralelo con el hilo principal del programa u otros hilos. Para ello
    #en el constructor se debe declarar una función que se ejecute en el nuevo hilo, sin poner sus
    #paréntesis.

    hiloVision = threading.Thread(target = visionRostrosAsistenteVirtual.reconocimientoFacial, args = (0, ))
    vozAsistenteVirtual.hablar("Hola di0, he activado mi reconocimiento facial, si no te reconozco preparete para
morir!! Ah y para cerrar la cámara da clic en la tecla r.")

    #threading.Thread().start(): Una vez creado el nuevo hilo con el constructor de la clase Thread, se
    #inicializará su ejecución con el método start().

    hiloVision.start()

    print("--Ok di_cer0, he activado mi reconocimiento facial, si no te reconozco preparete para moriir--")

#Si no se reconoce que el usuario diga ninguna de las palabras reservadas de arriba, el asistente
#virtual por medio de la librería de langchain y openai responde la pregunta hecha por el usuario.
else:

    #widgetAlarma.ejecutarAlarma(): Método de la clase propia widgetAlarma que ejecuta una alarma
    #que solamente puede ser detenida al presionar la tecla de SpaceBar.

    respuestaAsistenteVirtual, respuestaArchivo, respuestaMensaje =
cerebroAsistenteVirtual.preguntarChatbot(preguntaUsuario)

    vozAsistenteVirtual.hablar(respuestaAsistenteVirtual)

    print(respuestaAsistenteVirtual)

    #Si se reconoce que el usuario dice la palabra escribe o guarda, el asistente virtual por medio
    #de la clase propia widgetEscribirNotas escribe la respuesta dada al usuario en un archivo txt.
    #any():Función que recibe un bucle for que recorrerá un diccionario, lista o tupla para después
    #retornar un valor booleano True cuando al menos uno de sus elementos sea True o distinto a None.
    if any(palabra in preguntaUsuario for palabra in ["escribe", "guarda"]):
        vozAsistenteVirtual.hablar("Ok di0, voy a guardar la respuesta que te di en mi archivo de notas.")
        respuestaNotas = notasAsistenteVirtual.escribirNotaTxt(respuestaArchivo)
        vozAsistenteVirtual.hablar(respuestaNotas)
        print("-----Ok di_cer0, ya he escrito en mi archivo de notas como me dijiste...-----")

    #Si se reconoce que el usuario dice la palabra mensaje o Whatsapp, el asistente virtual por medio
    #de la clase propia widgetWhatsApp manda por mensaje la respuesta dada al usuario.
    if any(palabra in preguntaUsuario for palabra in ["mensaje", "whatsapp"]):
        vozAsistenteVirtual.hablar("Ok di0, voy a mandarte mi respuesta por mensaje a tu whatsapp.")
        respuestaWhatsApp = whatsappAsistenteVirtual.mandarMensaje(respuestaMensaje)
        vozAsistenteVirtual.hablar(respuestaWhatsApp)
        print("-----Ok di_cer0, ya he mandado el mensaje que me dijiste...-----")

#Condicional que está checando si la variable horaAlarma ya tiene asignada una hora de alarma.
if (horaAlarma != None):

```



```

        respuestaAlarma = alarmaAsistenteVirtual.sonarAlarma(horaAlarma)
        vozAsistenteVirtual.hablar(respuestaAlarma)
    except Exception as errorEjecucion:
        vozAsistenteVirtual.hablar("Lo siento, no escuché bien lo que dijiste debido a un error, repítelo porfa.")
        print("-----Lo siento di_cer0, no escuché bien lo que dijiste.-----")
        print(errorEjecucion)
        continue

#El límite de la duración de audio que puede procesar T.I.M.M.Y. en las instrucciones que se le manda es de 15
#segundos y el Mark I en sus respuestas tiene un delay de:

# - tiny: Delay de 2 segundos cuando le mando un comando hablado usando el comando.
# - base: Delay de 5 segundos cuando le mando un comando hablado usando el comando.
# - small: Delay de 10 segundos cuando le mando un comando hablado usando el comando.

```

Clase: oidoAsistente

```

#IMPORTACIÓN DE LIBRERÍAS:

import io          #io: Librería que permite la manipulación de los archivos y carpetas de nuestro ordenador.
#AudioSegment: Clase de la librería pydub que permite tomar el audio que perciba el micrófono del ordenador y
#luego que eso se pueda convertir en un archivo mp3 o wav.
import pydub       #pydub: Librería para procesar y exportar a un archivo datos de audio.
import speech_recognition as sr #sr: Librería que permite recabar audio del micrófono en tiempo real.
import whisper     #whisper: Librería de OpenAI que permite hacer la transcripción de audio o video a texto.
import tempfile    #tempfile: Librería que crea y maneja archivos temporales.
import os          #os: Librería que permite acceder a funciones y métodos relacionados con el sistema operativo.

#CREACIÓN DE ARCHIVO TEMPORAL:

#tempfile.mkdtemp(): El método mkdtemp() crea un directorio temporal en la carpeta predeterminada del sistema
#operativo, la cual es llamada temp y es una ubicación en el sistema operativo donde se almacenan archivos
#temporales generados por diversas aplicaciones y procesos.
archivoTemporal = tempfile.mkdtemp()

#os.path.join(): El método join() se utiliza para unir varios componentes de ruta (variables y constantes) en
#una sola ruta. Esto se utilizará para crear el archivo wav o mp3 que guardará un pedazo de la instrucción del
#usuario. Cabe mencionar que WAV es un formato de archivo de audio sin pérdidas, lo que significa que todos los
#datos de audio originales se conservan. Esto hace que los archivos WAV sean de mayor calidad que los archivos
#MP3, pero también los hace más grandes y pesados.
rutaArchivo = os.path.join(archivoTemporal, 'audioTemporal.wav')
print("Esta es la ruta del archivo temporal de las instrucciones recibidas en audio:\n" + str(rutaArchivo))

#RECONOCIMIENTO DE LAS INSTRUCCIONES DADAS AL USUARIO:

#speech_recognition.Recognizer(): El objeto Recognizer() se utiliza para crear un reconocedor de voz, el cual
#puede utilizarse para reconocer instrucciones dadas al asistente virtual por medio del habla humana.
listenerInstrucciones = sr.Recognizer()

#EscucharMicrofono: Clase propia para escuchar las instrucciones del usuario por medio del micrófono y almacenar
#eso en un archivo temporal, luego se transcribirá ese archivo de audio a texto a través del modelo Whisper y

```



```

#finalmente el texto resultante será transformado a minúsculas para que sea interpretado por el programa.
class EscucharMicrofono:

    #POO: En Python cuando al nombre de una función se le ponen dos guiones bajos antes de su nombre es porque
    #se está refiriendo a un método privado, es una buena práctica de sintaxis.

    #__escuchaInstrucciones(textoAsistente): Función propia y con modificador de acceso privado que se encarga
    #de escuchar y reconocer el audio por medio del micrófono del ordenador las palabras dichas por el usuario,
    #esto se tarda un poco en cargar la primera vez que se corre el programa debido a que el método
    #speech_recognition.Recognizer().listen() necesita un tiempo para inicializarse y configurarse; si se está
    #utilizando una computadora con poca potencia o micrófonos de baja calidad, es posible que se necesite
    #esperar más tiempo.

    def __escuchaInstrucciones(self):

        #MANEJO DE EXCEPCIONES: Es una parte de código que se conforma de dos partes, try y except:

        # - Primero se ejecuta el código que haya dentro del try y si es que llegara a ocurrir una excepción
        #   durante su ejecución, el programa brinca al código del except.

        # - En la parte de código donde se encuentra la palabra reservada except, se ejecuta cierta acción
        #   cuando ocurra el error esperado.

        #Se utiliza esta arquitectura de código cuando se quiera efectuar una acción donde se espera que pueda
        #ocurrir un error durante su ejecución.

        try:

            #with as source: La instrucción with se utiliza para definir una variable que tiene asignada un
            #método o recurso específico, el cual puede ser un archivo, una conexión a una base de datos, la
            #cámara, micrófono o cualquier otro objeto que requiera ser cerrado. La palabra clave as se utiliza
            #para asignar dicho recurso a una variable que puede utilizarse para acceder al recurso dentro del
            #bloque with.

            #Dentro del manejo de excepciones la instrucción with as source: asegura que se cierre el recurso,
            #incluso si se produce un error dentro del bloque try.

            #speech_recognition.Microphone(): El objeto Microphone puede utilizarse para grabar audio.

            with sr.Microphone() as microfono:

                print("\n\n-----Hola soy T.I.M.M.Y. Mark I tu asistente virtual, en que te puedo ayudar?...-----")

                #speech_recognition.Recognizer().adjust_for_ambient_noise(): El método adjust_for_ambient_noise()
                #sirve para aplicar un filtro a la señal de sonido que quite el ruido de fondo recibido en su
                #parámetro.

                listenerInstrucciones.adjust_for_ambient_noise(microfono)

                print("Quitando ruido de fondo... ya puedes hablar.")

                #speech_recognition.Recognizer().listen(): El método listen() sirve para poder escuchar de una
                #fuente de audio en tiempo real y convertirlo a texto, para ello primero se tuvo que haber
                #instanciado la clase Recognizer.

                textoInstruccionUsuario = listenerInstrucciones.listen(microfono)

                print("Microfono encendido y escuchando...")

                #io.BytesIO(): Método que sirve para crear un flujo de bytes en memoria que puede utilizarse
                #para almacenar datos binarios, como imágenes, audio y video.

                #speech_recognition.Recognizer().listen().get_wav_data(): El método get_wav_data() se utiliza
                #para almacenar el texto recibido en el micrófono del ordenador en un flujo de datos binarios
                #que se puedan guardar en un archivo de audio con extensión wav.

                datosAudio = io.BytesIO(textoInstruccionUsuario.get_wav_data())

```



```

#pydub.AudioSegment().from_file(): El método from_file() perteneciente a la clase AudioSegment
#se utiliza para cargar solamente los datos de audio donde se escuche la voz del usuario,
#logrando así que no se haga nada cuando el usuario está en silencio.
archivoAudio = pydub.AudioSegment.from_file(datosAudio)
print("Mandando texto del microfono a un archivo de audio wav...")
#pydub.AudioSegment().from_file().export(path, file_type): Lo que hace el método export() es
#exportar un segmento de audio a un archivo.
archivoAudio.export(rutaArchivo, format = 'wav')
print("Texto del microfono guardado en un archivo de audio temporal tipo wav...")

#Para identificar el tipo de excepción que ha ocurrido y utilizarlo en la instrucción except, se puede
#utilizar la clase Exception, que es una clase incorporada en Python utilizada para describir todos los
#tipos de excepciones, luego de colocar el nombre de la clase Exception se usa la palabra reservada "as"
#seguida de un nombre de variable, esto nos permitirá acceder a la instancia de la excepción y
#utilizarla dentro del except.
except Exception as error:
    print(error)

#Se retorna la ruta del archivo de la función porque de ahí extraerá el archivo de audio el modelo
#Whisper para procesarlo.
return rutaArchivo

#__transcripcionWhisper(): Función propia y con modificador de acceso privado que se encarga de recibir el
#archivo de audio recortado que contiene la instrucción mandada al asistente virtual para pasársela al
#modelo de Whisper para que la transcriba a texto.
def __transcripcionWhisper(self, audioRecortado):
    #whisper.load_model(): Con el método load_model() se pueden cargar los diferentes modelos de lenguaje de
    #la librería whisper, si se elige un modelo muy simple, se tendrá un peor desempeño, pero mientras más
    #complejo sea el modelo, más se tardará en procesar el audio, a esto dentro de la librería se le llama
    #Parameter y Relative Speed, donde a un mayor número de parámetros, se tendrá mayor capacidad de
    #aprendizaje en el modelo y la velocidad relativa indica la rapidez con la que el modelo puede generar
    #texto, esta se calcula dividiendo el tiempo que tarda el modelo en generar una palabra entre el número
    #de palabras que genera. Las características de los modelos disponibles son las siguientes:
    # - tiny:   Parameter = 39M,   Memoria que consume = 1GB   y Relative Speed = 32x.
    # - base:   Parameter = 74M,   Memoria que consume = 1GB   y Relative Speed = 16x.
    # - small:  Parameter = 244M,  Memoria que consume = 2GB   y Relative Speed = 6x.
    # - medium: Parameter = 769M,  Memoria que consume = 5GB   y Relative Speed = 2x.
    # - large:  Parameter = 1550M, Memoria que consume = 10GB  y Relative Speed = 1x.
    #Cuando el programa sea ejecutado por primera vez empezará a descargar el modelo para poderlo utilizar,
    #este proceso tarda un poco en terminar, pero después identificará por sí solo el lenguaje y las
    #palabras dichas en el audio, no importando si este tenga ruido o si es de alguna canción con
    #instrumentos detrás.
    Modelo = whisper.load_model("small")
    #whisper.load_model().transcribe(): El método transcribe() sirve para escuchar el audio de un archivo y
    #transcribirlo a texto a través del modelo elegido con el método load_model(). El proceso de carga tarda
    #un poco y el tiempo que el modelo se tarda en procesar el audio depende de su duración. Los parámetros
    #que recibe el método son la ubicación del archivo de audio que quiere transcribir y el parámetro fp16

```



```

#que indica si el cálculo se realizará en formato de punto flotante de 16 bits (FP16), pero como en
#algunas CPUs no se admite esta funcionalidad, se coloca como False, por lo que se está utilizando punto
#flotante de 32 bits (FP32) en su lugar. Además cabe mencionar que el audio escuchado por el método será
#recortado en cachos de 30 segundos y estos serán igualmente divididos en pedazos.
TranscripcionAudio_a_Texto = Modelo.transcribe(audioRecortado, language = "spanish", fp16 = False)
print("Interpretando texto guardado en un archivo temporal de audio wav con el modelo Whisper...\t")
#whisper.load_model().transcribe()["text"]: Texto traducido por Whisper de un archivo de audio.
return TranscripcionAudio_a_Texto["text"]

#oidoAsistenteVirtual(): Método con modificador de acceso público que permite ejecutar los métodos privados
#__transcripcionWhisper() para transcribir un archivo de audio a texto y __escuchaInstrucciones() para
#escuchar las instrucciones del usuario por medio del micrófono y almacenar eso en un archivo temporal. El
#resultado del texto escuchado del usuario será transformado a minúsculas aplicándole el método lower().
#self: La instrucción self se utiliza para hacer referencia al objeto que se está manipulando cuando se instancia
#la clase. Por eso es que a través de la misma nomenclatura de un punto se accede a los distintos atributos y/o
#métodos con un objeto desde fuera de la clase.
def oidoAsistenteVirtual(self):
    return self.__transcripcionWhisper(self.__escuchaInstrucciones()).lower()

```

Clase: vozAsistente

```

#IMPORTACIÓN DE LIBRERÍAS:
import pyttsx3 #pyttsx3: Biblioteca que sirve para generar una voz disponible en el sistema operativo.

#VOZ UTILIZADA POR EL ASISTENTE VIRTUAL:
#pyttsx3.init(): El método pyttsx3.init() se utiliza para inicializar el motor de texto a voz, el cual dará la
#habilidad de hablar al asistente virtual, convirtiendo el texto retornado por el modelo de Whisper en audio que
#salga de la computadora.
motorVoz = pyttsx3.init()
#pyttsx3.init().getProperty("voices"): El método getProperty() se utiliza para obtener una lista de las voces
#disponibles en el motor de texto a voz incluidos en el sistema operativo del ordenador.
voz_AsistenteVirtual = motorVoz.getProperty("voices")
#Bucle for para mostrar todas las opciones de sintetizadores de voz disponibles en el sistema operativo, el
#número de opciones dependerá de los lenguajes que pueda manejar el sistema operativo de la computadora que
#digan text to speech y se pueden ver al ingresar a la opción de: Windows -> Configuración -> Hora e Idioma ->
#Idioma y Región -> Idioma -> Idiomas que diga texto a voz. Si se quiere se podría agregar más idiomas.
for i in range(len(voz_AsistenteVirtual)):
    print(voz_AsistenteVirtual[i])
#pyttsx3.init().setProperty(): Este método toma dos parámetros y no devuelve ningún valor porque indica las
#siguientes características del sintetizador de voz del asistente virtual:
# - name: El nombre de la propiedad que se desea establecer.
# - rate: Indica la velocidad de la voz en palabras por minuto.
# - volume: El volumen de la voz, de 0 a 1.
# - voices: Lista de todas las voces disponibles.
# - voice: La voz que se utilizará para hablar.

```




```
# - langauge: El idioma que se utilizará para hablar.
# - engine: Controlador de voz que se utilizará.
# - debug: Variable booleana que indica si se debe habilitar el modo de depuración o no.
# - value: El valor de la propiedad que se desea establecer.
motorVoz.setProperty("rate", 190)
motorVoz.setProperty("voice", voz_AsistenteVirtual[3].id)

#vozWindows: Clase propia que utiliza una de las voces incluidas en los lenguajes descargados en el sistema
#operativo Windows para lograr así que el asistente virtual hable.
class vozWindows:
    #hablar(textoAsistente): Función propia que permite al asistente virtual hablar por medio de la bocina del
    #ordenador.
    def hablar(self, textoAsistente):
        #pyttsx3.init().say(): El método say() se utiliza para convertir texto en audio y emitirlo a través del
        #altavoz del sistema.
        motorVoz.say(textoAsistente)
        #pyttsx3.init().runAndWait(): El método runAndWait() bloquea la ejecución del programa hasta que el motor
        #de texto a voz haya terminado de hablar.
        motorVoz.runAndWait()
```

Clase: cerebroLangchainAsistente

```
#IMPORTACIÓN DE LIBRERÍAS:
#1.-MODELOS (Models): El modelo se refiere a la red neuronal que se va a utilizar para procesar el texto de entrada y
#generar una respuesta, los Large Language Model (LLM) responden preguntas sin guardar un historial, mientras que los
#Chats si guardan las preguntas y respuestas realizadas para crear una conversación. Existen varios modelos dentro de
#una misma compañía, por ejemplo, OpenAI cuenta con gpt3, gpt4, gpt3.5 turbo, etc.
print("-----LANGCHAIN-----")
print("-----1.-MODELOS-----")
#ChatOpenAI: Clase de la librería langchain que permite utilizar el modelo de chat (ChatGPT) de OpenAI con Python, este
#puede contestar preguntas adoptando un rol y guardar un historial durante la conversación.
from langchain.chat_models import ChatOpenAI #ChatOpenAI: Modelo de Chat.

#2.-PROMPTS: Es el texto que se le envía al modelo para generar una respuesta y en este es donde se utilizan las
#técnicas de Prompt Engineering, para ello la librería LangChain cuenta con diferentes clases que permiten utilizar
#dichas técnicas, dependiendo de si se está mandando el Prompt a un LLM o a un Chat.
print("-----2.-PROMPTS-----")
#PromptTemplate: Clase de la librería langchain que permite mandar instrucciones o preguntas personalizadas a un modelo
#LLM (Large Language Model) previamente invocado con Python, que no guarda un historial.
from langchain import PromptTemplate #PromptTemplate: Pregunta mandada a un modelo LLM.
#ChatPromptTemplate: Clase de la librería langchain que permite mandar instrucciones o preguntas personalizadas a un
#modelo de Chat, este puede contestar preguntas adoptando un rol a través de las siguientes clases:
# - SystemMessagePromptTemplate: Con esta clase se indica el rol que interpretará ChatGPT al responder las preguntas
# del usuario.
# - HumanMessagePromptTemplate: Con esta clase se representa el rol del usuario que manda preguntas a ChatGPT.
```



```

from langchain.prompts import ChatPromptTemplate #ChatPromptTemplate: Instrucciones mandadas a un modelo de chat.
from langchain.prompts import SystemMessagePromptTemplate, HumanMessagePromptTemplate

#4.-CADENAS (Chains): Con esta herramienta se permite enlazar un modelo con un Prompt, también con ella se pueden
#conectar varios modelos entre sí, hasta cuando son de distintos tipos, permitiéndonos así realizar varias iteraciones
#entre modelos durante una consulta para obtener un mejor procesamiento final de los datos cuando este se busca aplicar
#a tareas muy complejas.

print("-----4.-CADENAS-----")

#PROCESAMIENTO DE LLM: Permite encadenar un prompt con varios modelos de forma secuencial, uniendo así varias cadenas.
# - TransformChain: Con esta clase se implementa una cadena de transformación, que se aplica a una entrada para
# producir una salida con un formato personalizado. Las transformaciones pueden ser representadas por cualquier
# función que tome una secuencia como entrada y devuelva una secuencia como salida.

#Por lo tanto, para que una cadena TransformChain funcione, se debe declarar una función propia que cambie el formato
#de la salida de otra cadena.

from langchain.chains import TransformChain #TransformChain: Librería que crea una cadena de cadenas.

class cerebro_OpenAI:

    #CONSTRUCTOR O INICIALIZADOR DE LA CLASE: En él se declaran los parámetros que recibe la clase, que además se
    #utilizarán en los demás métodos, estos a fuerza deben tener un valor.

    #self: La instrucción self se utiliza para hacer referencia al objeto que se está manipulando cuando se instancia
    #la clase. Por eso es que a través de la misma nomenclatura de un punto se accede a los distintos atributos y/o
    #métodos con un objeto desde fuera de la clase.

    def __init__(self, parametro_de_la_clase):
        #ChatGPT API key
        self.LlaveApi = parametro_de_la_clase

    #POO: En Python cuando al nombre de una función se le ponen dos guiones bajos antes de su nombre es porque
    #se está refiriendo a un método privado, es una buena práctica de sintaxis.

    #__ChatbotLangchain(preguntaUsuario): Función propia y con modificador de acceso privado que se encarga de recibir
    #el texto de la pregunta realizada al modelo de Chat de OpenAI para contestar una pregunta según un rol asignado.
    def __ChatbotLangchain(self, preguntaUsuario):
        #ChatOpenAI(): En el constructor de la clase ChatOpenAI del paquete chat_models de la librería langchain se
        #indica:

        # - model_name: Parámetro que indica el modelo que se quiere utilizar, en este caso se utilizará gpt-3.5-turbo
        # que pertenece a GPT-3.5.

        # - openai_api_key: Con este parámetro se proporciona la API key, que por buenas prácticas debe provenir de
        # otro archivo.

        # - prompt_length: Longitud del prompt.

        # - max_tokens: Número máximo de tokens que se pueden generar.

        # - stop_token: El token de parada.

        # - temperature: La temperatura es un valor entre 0 y 1 que indica la creatividad con la que contesta el LLM,
        # si es demasiado grande, puede responder con algo totalmente aleatorio y si es muy bajo responderá lo mismo
        # siempre, función que podría ser deseada cuando por ejemplo se contestan problemas matemáticos.

        #Todos los modelos disponibles para usarse con OpenAI estan enlistados en el siguiente enlace y cada uno es
        #mejor en ciertas funciones que el otro:

```



```

#https://platform.openai.com/docs/models
openaiChatGPT = ChatOpenAI(model_name = "gpt-3.5-turbo", openai_api_key = self.LlaveApi, temperature = 0.4)#Chat.

#SYSTEM - ROL DEL CHAT AL RESPONDER PREGUNTAS DEL USUARIO: Para ello se utiliza un objeto PromptTemplate.
#PromptTemplate(): En el constructor de la clase PromptTemplate perteneciente a la librería langchain se
#indica:
# - template: Parámetro que indica la pregunta del prompt.
# - input_variables: Indica a través de una lista todos los nombres de las variables incluidas en la plantilla
#   del prompt, que se declararon dentro del template entre llaves {}.
plantillaPromptSistema = PromptTemplate(
    template = "Tu nombre es Timmy, eres una asistente virtual carismática, burlona y experta en chistes de Chihuahueros."
+
    "Puedes contestar las preguntas del usuario espontáneamente de forma sarcástica y otras veces de forma
" +
    "seria y lógica para resolver sus problemas. El nombre del usuario es di0." +
    "Tu como asistente virtual puedes establecer alarmas, escribir en tu archivo NotasTimmy.txt, reproducir
" +
    "videos en YouTube, mandar mensajes por WhatsApp, mostrar imagenes con mapas de rutas, crear eventos
en " +
    "calendarios de Google Calendar y utilizar visión artificial para identificar colores o reconocimiento
" +
    "facial.",
    input_variables = []
)
#SystemMessagePromptTemplate(): Esta clase recibe como parámetro un objeto PromptTemplate, que previamente ya
#tiene diseñado el template que se mandará en el Prompt, indicándole al Chat el rol que está interpretando al
#responder.
promptSistema = SystemMessagePromptTemplate(prompt = plantillaPromptSistema)

#HUMAN - PREGUNTAS QUE EL USUARIO LE HACE AL MODELO: Para ello se utiliza un objeto PromptTemplate.
plantillaPromptHumano = PromptTemplate(
    template = "Con tu basto conocimiento que te vuelve una genio contesta la siguiente pregunta del usuario:\n"
    "{pregunta}",
    input_variables = ["pregunta"]
)
#HumanMessagePromptTemplate(): Esta clase recibe como parámetro un objeto PromptTemplate, que previamente ya
#tiene diseñado el template de la pregunta que hace el usuario al chat.
promptHumano = HumanMessagePromptTemplate(prompt = plantillaPromptHumano)

#ChatPromptTemplate.from_messages(): Método que sirve para unificar los templates previamente creados para el
#sistema (que le dice al modelo el rol que debe interpretar al responder mis preguntas), para el humano (que
#indica tal cual la pregunta realizada por el usuario) y de la AI (que es un rol adoptado por el modelo para
#guardar las preguntas y respuestas realizadas en un historial), creando así una conversación. El parámetro que
#recibe el método es una lista que incluye todas las plantillas de Prompt mencionadas previamente.
plantillaChatPrompt = ChatPromptTemplate.from_messages([promptSistema, promptHumano])

```



```

#ChatPromptTemplate().format_prompt().to_messages(): Método que rellena las variables del template mandado al
#Chat con valores de entrada para el prompt del sistema, del humano y de la AI, retornando una lista.
promptMandadoChat = plantillaChatPrompt.format_prompt(pregunta = preguntaUsuario).to_messages()
#str(): Método que convierte un número, lista, diccionario, etc. en un string para que pueda ser impreso en
#consola.
print("Personalidad de Chatbot:\n" + str(promptMandadoChat[0].content) + "\n\n" +
      "Pregunta hecha al Chatbot:\n" + str(promptMandadoChat[1].content) + "\n")
#ChatOpenAI(ChatPromptTemplate().format().to_messages()): Prompt mandado al modelo de Chat.
respuestaChat = openaiChatGPT(promptMandadoChat)
#Del diccionario retornado, el key de content es el que contiene la respuesta de la pregunta.
print("Respuesta del Chatbot:\n", respuestaChat.content + "\n\n")
return respuestaChat.content

#__formatoHablarRespuesta(respuestaChat): Función propia y con modificador de acceso privado que se encarga de
#recibir la respuesta proporcionada por el modelo de Chat de OpenAI, para luego a través de una cadena de formato
#llamada TransformChain(), quitar los saltos de línea del texto para que pueda hablar de forma fluida el asistente
#virtual.
def __formatoHablarRespuesta(self, respuestaChat):
    #Función propia que cambia el formato de cualquier salida proporcionada por un modelo de Chat o LLM.
    def eliminarSaltosDeLinea(entrada):
        #Función que intercambia los saltos de línea por espacios.
        texto = entrada["texto"] #Recibe una lista con un diccionario interno de key = texto.
        #lista.replace(): Método que reemplaza dentro de una lista un string por otro.
        return {"texto_limpio" : texto.replace("\n", " ")}
    #TransformChain(): Objeto que recibe un prompt, cambia su formato de una forma personalizada y lo retorna en
    #una variable nueva.
    # - input_variables: Indica a través de una lista los prompts de entrada.
    # - output_variables: Indica a través de una lista el nombre de la variable de salida ya con el formato
    # deseado.
    # - transform: Recibe el nombre de la función propia que transforma el formato de la variable de entrada.
    cadenaTransformarFormato = TransformChain(input_variables = ["texto"],
                                              output_variables = ["texto_limpio"],
                                              transform = eliminarSaltosDeLinea)
    transformChainHablar = cadenaTransformarFormato.run(respuestaChat)
    return transformChainHablar

#__formatoMensajeRespuesta(respuestaChat): Función propia y con modificador de acceso privado que se encarga de
#recibir la respuesta proporcionada por el modelo de Chat de OpenAI, para luego a través de una cadena de formato
#llamada TransformChain() transformar la salida en una lista, con el fin de que el formato de la respuesta que se
#mandará por mensaje a WhatsApp se respete.
def __formatoMensajeRespuesta(self, respuestaChat):
    def listaSaltosDeLinea(entrada):
        texto = entrada["texto"]
        return {"texto_lista_mensaje" : texto.split("\n")}
    cadenaTransformarFormato = TransformChain(input_variables = ["texto"],

```



```

        output_variables = ["texto_lista_mensaje"],
        transform = listaSaltosDeLinea)

transformChainMensaje = cadenaTransformarFormato.run(respuestaChat)
return transformChainMensaje

#preguntarChatbot(preguntaUsuario): Método con modificador de acceso público que permite ejecutar los métodos
#privados __ChatbotLangchain(), __formatoHablarRespuesta y __formatoMensajeRespuesta para procesar la pregunta
#hecha por el usuario y responder en forma de lista que separa las líneas de texto de la respuesta y en forma de
#texto sin saltos de línea para que el asistente virtual pueda decir la respuesta y además la pueda mandar por
#medio de Whatsapp y que su formato se respete.
def preguntarChatbot(self, preguntaUsuario):
    respuestaArchivo = self.__ChatbotLangchain(preguntaUsuario) #Respuesta del método __ChatbotLangchain.
    respuestaHablar = self.__formatoHablarRespuesta(respuestaArchivo) #Formato de respuesta sin saltos de línea.
    respuestaMensaje = self.__formatoMensajeRespuesta(respuestaArchivo) #Formato de respuesta en forma de lista.
    #return variable_1, variable_2, ..., variable_n: Cuando después de un método return se tiene más de 1
    #variable, al ejecutar el método se deberán declarar dos variables que reciban cada resultado de forma
    #correspondiente a través de la siguiente sintaxis:
    # variable_1, variable_2, variable_n = objetoDeClase.métodoVariasVariables()
    return respuestaHablar, respuestaArchivo, respuestaMensaje

```

Clase: alarmaAsistente

```

#ESTABLECER ALARMA:
import datetime #datetime: Librería que proporciona método para trabajar con fechas y horas en Python.
#re: Librería que identifica patrones en expresiones regulares. Una expresión regular es un patrón de texto que se
#puede utilizar para buscar, reemplazar o extraer texto de una cadena.
import re #re: Librería que permite utilizar expresiones regulares para detectar patrones.
import keyboard #keyboard: Librería que permite detectar las pulsaciones de teclas en tu teclado
import pygame #pygame: librería de multimedia que permite mostrar gráficos, reproducir sonido, etc.

class widgetAlarma:
    def programarAlarma(self, preguntaUsuario):
        #.replace().lower(): Lo que hace el método replace() es reemplazar todas las palabras que aparezcan en un
        #string en otra cadena específica y el método lower() sirve para convertir todas las letras de una cadena en
        #minúsculas, esto es importante hacerlo porque al reproducir una canción o mandar una búsqueda a internet,
        #siempre es mejor tener puras minúsculas.
        instruccionHora = preguntaUsuario.replace(' alarma', '').lower()
        #strip(): Método utilizado para eliminar los caracteres especificados en una cadena, como espacios saltos de
        #línea (\n), tabuladores (\t), etc. Si no se le pasa ningún parámetro, eliminará los espacios del string.
        instruccionHora = instruccionHora.strip()
        print(instruccionHora)
        #re.findall(r): El método findall() utilizado para encontrar todas las ocurrencias de una expresión regular en
        #una cadena de texto (string) recibe 3 parámetros, la expresión regular que se quiere encontrar, la cadena de
        #caracteres donde se buscará y modificadores opcionales que se pueden utilizar para personalizar el
        #comportamiento de la búsqueda.

```



```

# - pattern: Parámetro que indica la expresión regular utilizada para extraer partes específicas del string.
# Las expresiones regulares más comunes son:
# - Búsqueda de números en un string:
#   - \d: Expresión regular que busca un dígito del 0 al 9 en un string.
#   - \d+: Expresión regular que coincide con uno o más dígitos del 0 al 9.
#   - \d{2,4}: Expresión regular que coincide con una secuencia de dígitos de 2 al 4 específicamente,
#     o sea 234.
# - Búsqueda de letras en un string:
#   - \w: Expresión regular que busca una letra mayúscula o minúscula, un dígito del 0 al 9 o un guión
#     bajo.
#   - \w+: Expresión regular que coincide con uno o más caracteres de los descritos anteriormente.
#   - [A-Za-z]: Expresión regular que busca solo una letra mayúscula o minúscula.
#   - \b: Expresión regular que representa el inicio o final de una palabra en un string, usualmente se
#     incluyen dos para indicar que se debe buscar un patrón en cada palabra. Por ejemplo, si se declara la
#     expresión regular \bHola\b, esto extraerá todas las palabras Hola dentro de un string.
# - Búsqueda de palabras en un string (grupos de captura):
#   - (patrón1|patrón2): Expresión regular que crea un grupo de captura para identificar uno o varios
#     patrones específicos.
#   - |: Compuerta OR utilizada para reconocer uno o varios patrones dentro de un grupo de captura.
# - Búsqueda de espacios en blanco, puntos y paréntesis en un string:
#   - \s: Expresión regular que coincide con un carácter de espacio vacío, como un espacio, tabulador (\t),
#     salto de línea (\n), etc.
#   - \.: Expresión regular que encuentra un punto en un string.
#   - \( : Expresión regular que encuentra un paréntesis de apertura en un string.
#   - \): Expresión regular que encuentra un paréntesis de cierre en un string.
# - Búsqueda de repeticiones en un string:
#   - *: Encuentra cero o más repeticiones de la expresión regular que tenga a la izquierda.
#   - +: Encuentra una o más repeticiones de la expresión regular que tenga a la izquierda.
#   - ?: Encuentra cero o una repetición de la expresión regular que tenga a la izquierda. Esto significa
#     que la expresión regular que la precede es opcional y puede aparecer una vez o no aparecer en
#     absoluto en la cadena que se está buscando.
#   - {2,4}: Coincide con 2, 3 o 4 repeticiones de la expresión regular que tenga a la izquierda.
# - Anclaje de patrones:
#   - ^: Cuando se coloca al principio de un patrón indica que la coincidencia debe encontrarse al comienzo
#     de la línea de texto. Por ejemplo, si se declara la expresión regular ^Hola, esto será cierto solo
#     cuando la palabra Hola se encuentre al inicio del string.
#   - $: Cuando se coloca al final de un patrón indica que la coincidencia debe encontrarse al final de la
#     línea de texto. Por ejemplo, si se declara la expresión regular $mundo, esto será cierto solo cuando
#     la palabra mundo se encuentre al final del string.
#   - ?: : Cuando se coloca un signo de interrogación seguido de dos puntos se está indicando que es un
#     grupo no capturador, esto significa que se tomará en cuenta para hacer coincidencias, pero no será
#     incluido dentro de la lista de tuplas.
# - string: Palabra en la que se desea buscar patrones a través de la expresión regular especificada.
# - flags: Modificadores opcionales que personalizan el comportamiento de la búsqueda.
#   - re.IGNORECASE: Realiza la búsqueda sin distinción entre mayúsculas y minúsculas.

```



```

# - re.MULTILINE: Permite que el patrón coincida con múltiples líneas en la cadena.
# - re.DOTALL: Hace que el carácter . en el patrón coincida también con el carácter de nueva línea \n.
#Esta expresión regular busca números en un string.
numeroshora = re.findall(r"\d+", instruccionHora)
print(numeroshora)

#join(): Este método toma una lista de strings como argumento y los concatena, juntando cada string con el
#carácter especificado.
hora = ":".join(numeroshora)
print(hora)

#Para que el string que representa la hora pueda ser convertido a un objeto datetime con formato de 12
#horas, se debe indicar si la hora de este es pm o am, pero para que esto sea entendido por el método
#strptime(), se debe indicar en mayúsculas.
if("pm" in instruccionHora):
    hora += " PM"
else:
    hora += " AM"

#datetime.datetime.strptime(): Método de la clase datetime que toma una cadena que representa una fecha y
#hora en un formato específico y la convierte a un objeto datetime, para ello recibe las siguientes
#instrucciones en su segundo parámetro:
# - %H: Hora en formato de 24 horas (00-23).
# - %I: Hora en formato de 12 horas (01-12). Para esto se debe indicar si la hora es AM o PM.
# - %p: AM o PM (funciona con %I para indicar la parte de la tarde o la mañana).
# - %M: Minutos (00-59).
# - %S: Segundos (00-59).
# - %f: Microsegundos (000000-999999).
# - %z: Desplazamiento de la zona horaria UTC en formato ±HHMM o ±HH:MM.
hora = datetime.datetime.strptime(hora, "%I:%M %p") #Se indica "%I" para usar el formato de 12 horas.
#Conversión de objeto datetime con formato de 12 horas en formato de 24 horas. Esto se hace para que sea
#compatible con la hora actual retornada por medio del método datetime.datetime.now().strftime("%H:%M").
horaAlarma = hora.strftime("%H:%M")
print(horaAlarma)

#datetime.datetime.now().strftime(): Método que devuelve la fecha y hora actual en el formato
#YYYY-MM-DD HH:MM:SS, al cual se accede con el string "%Y-%m-%d %H:%M:%S".
print(datetime.datetime.now().strftime("%H:%M"))

return horaAlarma

def sonarAlarma(self, horaAlarma):
    respuestaAlarma = ""
    if (datetime.datetime.now().strftime("%H:%M") == horaAlarma):
        while True:
            print("-----Hola di_cer0, ya es la hora en la que me pediste que estableciera una alarma-----")
            #pygame.mixer.init(): El método init() inicializa el objeto mixer de la librería pygame
            #que sirve para reproducir sonidos con Python.
            pygame.mixer.init()

```




```

        #pygame.mixer.music.load(): El método load() carga un archivo de música con formato wav,
        #mp3 u ogg en la memoria para que pueda ser reproducido al ejecutar el método play().
        pygame.mixer.music.load("C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/0.-Archivos_Ejercicios_Python/Till I Collapse - Eminem.mp3")

        pygame.mixer.music.play()

        #keyboard.read_key(): Método que devuelve la tecla presionada por el usuario.
        if(keyboard.read_key() == "space"):
            #pygame.mixer.music.stop(): Método que detiene la música que se haya reproducido
            #anteriormente con la función play().
            pygame.mixer.music.stop()

            break

        horaAlarma = None

        respuestaAlarma = "Hola di0, alarma detenida"

    return respuestaAlarma

```

Clase: notasAsistente

```

#ABRIR Y GUARDAR NOTAS EN UN ARCHIVO TXT:
import os          #os: Librería que permite acceder a funciones del sistema operativo, como abrir archivos.

class widgetEscribirNotas:
    def escribirNotaTxt(self, respuestaAsistenteVirtual):
        respuestaNotas = ""

        filename = "C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/NotasTimmy.txt"

        #MANEJO DE EXCEPCIONES: Es una parte de código que se conforma de dos partes, try y except:
        # - Primero se ejecuta el código que haya dentro del try y si es que llegara a ocurrir una excepción
        #   durante su ejecución, el programa brinca al código del except.
        # - En la parte de código donde se encuentra la palabra reservada except, se ejecuta cierta acción
        #   cuando ocurra el error esperado.

        #Se utiliza esta arquitectura de código cuando se quiera efectuar una acción donde se espera que pueda
        #ocurrir un error durante su ejecución.

        try:

            #with as source: La instrucción with se utiliza para definir una variable que tiene asignada un
            #método o recurso específico, el cual puede ser un archivo, una conexión a una base de datos, la
            #cámara, micrófono o cualquier otro objeto que requiera ser cerrado. La palabra clave as se utiliza
            #para asignar dicho recurso a una variable que puede utilizarse para acceder al recurso dentro del
            #bloque with.

            #Dentro del manejo de excepciones la instrucción with as source: asegura que se cierre el recurso,
            #incluso si se produce un error dentro del bloque try.

            #open(): Método que sirve para abrir un archivo cualquiera y manejarlo por medio de Python.
            with open(filename, "w", encoding="utf-8") as f:

                f.write(respuestaAsistenteVirtual)

            #var_file_open.close(): Método para cerrar un archivo previamente abierto con el
            #método open(), es peligroso olvidar colocar este método, ya que la computadora lo

```



```

        #considerará como si nunca hubiera sido cerrado, por lo cual no podré volver a
        #abrirlo al dar clic sobre él.

        f.close()

        #os.system(): Método que permite ejecutar comandos del sistema operativo desde
        #Python.

        # - ls: Mostrar todas las carpetas del directorio.

        # - cd: Command directory sirve para moverme de directorio.

        # - start: Abrir o ejecutar un archivo.

        # - ejecutable + archivo: Con este comando que incluye un ejecutable y un archivo
        # se indica al sistema operativo que abra cierto archivo usando un programa en
        # específico.

        programa_para_abrir = "notepad.exe"

        os.system(f'{programa_para_abrir} "{filename}"')

except FileNotFoundError as errorArchivo:

    #open(): Método que sirve para abrir un archivo cualquiera, para ello es necesario
    #indicar dos parámetros, el primero se refiere a la ruta relativa o absoluta del archivo
    #previamente creado y la segunda indica qué es lo que se va a realizar con él, el
    #contenido del archivo se asigna a una variable.

    # - w: Sirve para escribir en un archivo, pero borrará la información que previamente
    # contenía el archivo.

    # - a: Sirve para escribir en un archivo sin que se borre la info anterior del archivo,
    # se llama append.

    file = open(filename, "w", encoding="utf-8")
    file.write(respuestaAsistenteVirtual)
    file.close()

    #os.system(): Método que permite ejecutar comandos del sistema operativo desde Python.

    # - ls: Mostrar todas las carpetas del directorio.

    # - cd: Command directory sirve para moverme de directorio.

    # - start: Abrir o ejecutar un archivo.

    # - ejecutable + archivo: Con este comando que incluye un ejecutable y un archivo se
    # indica al sistema operativo que abra cierto archivo usando un programa en
    # específico.

    programa_para_abrir = "notepad.exe"

    os.system(f'{programa_para_abrir} "{filename}"')

print("-----Hola di_cer0, ya respondí tu pregunta y guardé la respuesta en mis notas...-----")

respuestaNotas = "Hola di0, ya escribí la respuesta que te dí en mi archivo de notas y luego te lo mostré."

return respuestaNotas

```

Clase: whatsappAsistente

```

#IMPORTACIÓN DE LIBRERÍAS:

import webbrowser    #webbrowser: Librería que permite abrir y utilizar navegadores web en Python.

import pyautogui      #pyautogui: Librería para controlar el mouse y teclado de la computadora con Python.

import time           #time: Librería del manejo de tiempos, como retardos, contadores, etc.

```

```

class widgetWhatsApp:
    #CONSTRUCTOR O INICIALIZADOR DE LA CLASE: En él se declaran los parámetros que recibe la clase, que además se
    #utilizarán en los demás métodos, estos a fuerza deben tener un valor.
    #self: La instrucción self se utiliza para hacer referencia al objeto que se está manipulando cuando se instancia
    #la clase. Por eso es que a través de la misma nomenclatura de un punto se accede a los distintos atributos y/o
    #métodos con un objeto desde fuera de la clase.
    def __init__(self, parametro_de_la_clase):
        #ChatGPT API key
        self.contacto = parametro_de_la_clase

    def mandarMensaje(self, mensaje):
        respuestaWhatsApp = ""
        #Bucle for que recorre la respuesta del modelo de chat de OpenAI que viene en forma de lista para mandar cada
        #línea de texto en un mensaje diferente, ya que no es posible de forma gratuita mandar todo el texto en un
        #solo mensaje respetando los saltos de línea del texto.
        for i in range(len(mensaje)):
            #Solo en la primera línea de texto se abre el navegador y se espera que se habra bien para mandar el
            #primer mensaje.
            if(i == 0):
                #webbrowser.open(): Método que abre un navegador web en la dirección URL especificada. El truco que
                #se realiza para mandar mensajes de whatsapp es que la URL de su sitio permite ingresar el mensaje y
                #teléfono de la persona a la que se busca mandar el WhatsApp.
                webbrowser.open(f"https://web.whatsapp.com/send?phone={self.contacto}&text={mensaje[i]}")
                #time.sleep(): Método que permite detener el programa cierto número de segundos.
                time.sleep(15)
                #pyautogui.press(): Método que simula presionar una tecla del teclado. El parámetro obligatorio que
                #recibe es el nombre de la tecla, que puede ser una cadena o un valor ASCII. El método también acepta
                #un parámetro opcional llamado interval, que especifica el intervalo de tiempo en segundos entre
                #presionar la tecla y soltarla.
                pyautogui.press("enter")
            #Los demás mensajes simplemente se mandan al introducir cada línea de mensaje en la página de Whatsapp
            #Web y presionar rápidamente la tecla de enter después de introducir cada línea del mensaje.
            else:
                pyautogui.write(mensaje[i])
                pyautogui.press("enter")
            #Finalmente se espera un tiempo y se cierra el navegador donde se encuentra abierta la pestaña.
            pyautogui.write("-----MENSAJE DE T.I.M.Y. RECIBIDO CORRECTAMENTE-----")
            pyautogui.press("enter")
            time.sleep(2)
        respuestaWhatsApp = "Hola di0, ya te mandé un Whats con mi respuesta a tu cel."
        return respuestaWhatsApp

```

Clase: googleCalendarAsistente

#IMPORTACIÓN DE LIBRERÍAS:



```

import datetime #datetime: Librería que proporciona método para trabajar con fechas y horas en Python.
import time     #time: Librería del manejo de tiempos, como retardos, contadores, etc.
import re       #re: Librería que permite utilizar expresiones regulares para detectar patrones.
import locale   #locale: Librería que permite obtener la locación actual del usuario.
import webbrowser #webbrowser: Librería que permite abrir y utilizar navegadores web en Python.

class widgetGoogleCalendar:

    #CONSTRUCTOR O INICIALIZADOR DE LA CLASE: En él se declaran los parámetros que recibe la clase, que además se
    #utilizarán en los demás métodos, estos a fuerza deben tener un valor.

    #self: La instrucción self se utiliza para hacer referencia al objeto que se está manipulando cuando se instancia
    #la clase. Por eso es que a través de la misma nomenclatura de un punto se accede a los distintos atributos y/o
    #métodos con un objeto desde fuera de la clase.

    def __init__(self, parametro_de_la_clase):
        #ChatGPT API key
        self.servicioAPI = parametro_de_la_clase

    #POO: En Python cuando al nombre de una función se le ponen dos guiones bajos antes de su nombre es porque
    #se está refiriendo a un método privado, es una buena práctica de sintaxis.

    def __tituloEvento(self, preguntaUsuario):
        #re.findall(r): El método findall() utilizado para encontrar todas las ocurrencias de una expresión
        #regular en una cadena de texto (string) recibe 3 parámetros, la expresión regular que se quiere encontrar,
        #la cadena de caracteres donde se buscará y modificadores opcionales que se pueden utilizar para
        #personalizar el comportamiento de la búsqueda.

        # - pattern: Parámetro que indica la expresión regular utilizada para extraer partes específicas del string.
        # Las expresiones regulares más comunes son:

        # - Búsqueda de números en un string:
        #     - \d: Expresión regular que busca un dígito del 0 al 9 en un string.
        #     - \d+: Expresión regular que coincide con uno o más dígitos del 0 al 9.
        #     - \d{2,4}: Expresión regular que coincide con una secuencia de dígitos de 2 al 4 específicamente,
        #         osea 234.
        # - Búsqueda de letras en un string:
        #     - \w: Expresión regular que busca una letra mayúscula o minúscula, un dígito del 0 al 9 o un guión
        #         bajo.
        #     - \w+: Expresión regular que coincide con uno o más caracteres de los descritos anteriormente.
        #     - [A-Za-z]: Expresión regular que busca solo una letra mayúscula o minúscula.
        #     - \b: Expresión regular que representa el inicio o final de una palabra en un string, usualmente se
        #         incluyen dos para indicar que se debe buscar un patrón en cada palabra. Por ejemplo, si se declara
        #         la expresión regular \bHola\b, esto extraerá todas las palabras Hola dentro de un string.
        # - Búsqueda de palabras en un string (grupos de captura):
        #     - (patrón1|patrón2): Expresión regular que crea un grupo de captura para identificar uno o varios
        #         patrones específicos.
        #     - |: Compuerta OR utilizada para reconocer uno o varios patrones dentro de un grupo de captura.
        # - Búsqueda de espacios en blanco, puntos y paréntesis en un string:
        #     - \s: Expresión regular que coincide con un carácter de espacio vacío, como un espacio, tabulador
        #         (\t), salto de línea (\n), etc.

```



```

# - \.: Expresión regular que encuentra un punto en un string.
# - \(: Expresión regular que encuentra un paréntesis de apertura en un string.
# - \): Expresión regular que encuentra un paréntesis de cierre en un string.
# - Búsqueda de repeticiones en un string:
# - *: Encuentra cero o más repeticiones de la expresión regular que tenga a la izquierda.
# - +: Encuentra una o más repeticiones de la expresión regular que tenga a la izquierda.
# - ?: Encuentra cero o una repetición de la expresión regular que tenga a la izquierda. Esto
# significa que la expresión regular que la precede es opcional y puede aparecer una vez o no
# aparecer en absoluto en la cadena que se está buscando.
# - {2,4}: Coincide con 2, 3 o 4 repeticiones de la expresión regular que tenga a la izquierda.
# - Anclaje de patrones:
# - ^: Cuando se coloca al principio de un patrón indica que la coincidencia debe encontrarse al
# comienzo de la línea de texto. Por ejemplo, si se declara la expresión regular ^Hola, esto será
# cierto solo cuando la palabra Hola se encuentre al inicio del string.
# - $: Cuando se coloca al final de un patrón indica que la coincidencia debe encontrarse al final de
# la línea de texto. Por ejemplo, si se declara la expresión regular $mundo, esto será cierto solo
# cuando la palabra mundo se encuentre al final del string.
# - ?: : Cuando se coloca un signo de interrogación seguido de dos puntos se está indicando que es un
# grupo no captador, esto significa que se tomará en cuenta para hacer coincidencias, pero no será
# incluido dentro de la lista de tuplas.
# - string: Palabra en la que se desea buscar patrones a través de la expresión regular especificada.
# - flags: Modificadores opcionales que personalizan el comportamiento de la búsqueda.
# - re.IGNORECASE: Realiza la búsqueda sin distinción entre mayúsculas y minúsculas.
# - re.MULTILINE: Permite que el patrón coincida con múltiples líneas en la cadena.
# - re.DOTALL: Hace que el carácter . en el patrón coincida también con el carácter de nueva línea \n.
#Esta expresión regular extrae todas las palabras que se encuentren entre dos conjuntos de palabras clave.
patronTitulo = re.findall(r"\b(titulo de|título de|se
llama|tituladas|titulados|título|título|llamado|nombrado|nombrada|nombre)\b\s+(.??)(?:\s+\b(descripción|descripcion|fecha
|hora|inicio|inicial|final|fin)\b|$)", preguntaUsuario, re.IGNORECASE|re.UNICODE)

#Bucle for each de una sola línea para extraer la posición intermedia de la lista de tuplas.
tituloLista = [tupla[1] for tupla in patronTitulo]

#join(): Este método toma una lista de strings como argumento y los concatena, juntando cada string con el
#carácter especificado.
tituloEvento = ', '.join(tituloLista)

#any():Función que recibe un bucle for que recorrerá un diccionario, lista o tupla para después retornar un
#valor booleano True cuando al menos uno de sus elementos sea True o distinto a None.

#string.endswith(): El método endswith() aplicado a un string verifica si su último caracter es uno en
#específico y si es así retorna un valor booleano True.
if any(tituloEvento.endswith(sufijo) for sufijo in [",", "y", " ", " ", " y "]):
    #replace(): La función del método replace() es reemplazar todas las veces que una palabra aparezca en
    #un string por otra cadena específica.
    tituloEvento = tituloEvento.replace(",", " ")
    tituloEvento = tituloEvento.replace("y", " ")

#La instrucción not seguida de una variable en un condicional evalúa si esta es un string o lista vacía.
if not tituloEvento:

```



```

        tituloEvento = "Título no detectado, evento sin nombrar."
    return tituloEvento

def __descripcionEvento(self, preguntaUsuario):
    #Esta expresión regular extrae todas las palabras que se encuentren entre dos conjuntos de palabras clave.
    patronDescripcion = re.findall(r"(descripcion de|descripción
de|descripcion|descripción)\s+(.*?)\s+(?:\s+\b(titulo|título|fecha|hora|inicio|inicial|final|fin)\b|$)", preguntaUsuario,
re.IGNORECASE|re.UNICODE)

    #Bucle for each de una sola línea para extraer la posición intermedia de la lista de tuplas.
    descripcionLista = [tupla[1] for tupla in patronDescripcion]

    #join(): Este método toma una lista de strings como argumento y los concatena, juntando cada string con el
    #carácter especificado.
    descripcionEvento = ', '.join(descripcionLista)

    #La siguiente instrucción verifica si el último carácter es una "," o una "y" y si es así lo reemplaza por
    #un espacio vacío.
    if any(descripcionEvento.endswith(sufijo) for sufijo in [",", "y", " ", " ", " y "]):
        descripcionEvento = descripcionEvento.replace(",", " ")
        descripcionEvento = descripcionEvento.replace("y", " ")

    #La instrucción not seguida de una variable en un condicional evalúa si esta es un string o lista vacía.
    if not descripcionEvento:
        descripcionEvento = "Evento sin descripción."
    return descripcionEvento

#Para establecer un evento en Google Calendar la fecha y hora se debe proporcionar en formato ISO, lo cual es
#logrado a través de la librería datetime.

def __horaInicioEvento(self, preguntaUsuario):
    preguntaUsuario = preguntaUsuario.lower() #lower(): Transformar todas las letras a minúsculas.
    instruccionHora = preguntaUsuario.strip() #lower(): Elimina los caracteres en blanco del string.
    numeroshora = ""

    if (":" in instruccionHora):
        #Esta expresión regular extrae todos los números que se encuentren entre dos puntos y antes de las
        #palabras am o pm, pero como se utiliza la instrucción ?:, las palabras am o pm no serán guardadas en
        #una tupla dentro de la lista extraída, sino que solo serán extraídos y separados los números.
        numeroshora = re.findall(r"(\d+):(\d+) (?:am|pm)", instruccionHora, re.IGNORECASE|re.UNICODE)
    else:
        #Expresión regular que checa si el número llegó con un punto (.) en vez de dos (:).
        numeroshora = re.findall(r"(\d+)\.(\d+) (?:am|pm)", instruccionHora, re.IGNORECASE|re.UNICODE)

    #join(): Este método toma una lista o tupla de strings como argumento y los concatena, juntando cada string
    #con el carácter especificado.
    hora = ":".join(numeroshora[0])

    #Para que el string que representa la hora pueda ser convertido a un objeto datetime con formato de 12
    #horas, se debe indicar si la hora de este es pm o am, pero para que esto sea entendido por el método
    #strftime(), se debe indicar en mayúsculas.
    if("pm" in preguntaUsuario):
        hora += " PM"

```



```

else:
    hora += " AM"

#datetime.datetime.strptime(): Método de la clase datetime que toma una cadena que representa una fecha y
#hora en un formato específico y la convierte a un objeto datetime, para ello recibe las siguientes
#instrucciones en su segundo parámetro:
# - %H: Hora en formato de 24 horas (00-23).
# - %I: Hora en formato de 12 horas (01-12). Para esto se debe indicar si la hora es AM o PM.
# - %p: AM o PM (funciona con %I para indicar si el formato de la hora es en la tarde o la mañana).
# - %M: Minutos (00-59).
# - %S: Segundos (00-59).
# - %f: Microsegundos (000000-999999).
# - %z: Desplazamiento de la zona horaria UTC en formato ±HHMM o ±HH:MM.
hora = datetime.datetime.strptime(hora, "%I:%M %p") #Se indica "%I" para usar el formato de 12 horas.
#Conversión de objeto datetime con formato de 12 horas en formato de 24 horas. Esto se hace para que sea
#compatible con la hora actual retornada por medio del método datetime.datetime.now().strftime("%H:%M").
horaAlarma = hora.strftime("%H:%M")
return horaAlarma

def __fechaInicioEvento(self, preguntaUsuario):
    preguntaUsuario = preguntaUsuario.lower() #lower(): Transformar todas las letras a minúsculas.
    #El diccionario meses realiza la conversión de la palabra de cada mes en su equivalente numérico.
    meses = {
        'enero': '01',
        'febrero': '02',
        'marzo': '03',
        'abril': '04',
        'mayo': '05',
        'junio': '06',
        'julio': '07',
        'agosto': '08',
        'septiembre': '09',
        'octubre': '10',
        'noviembre': '11',
        'diciembre': '12'
    }

    #Esta expresión regular extrae 3 palabras que se encuentren antes y después de ciertas palabras clave.
    #diccionario.keys(): Los diccionarios se componen de llaves y valores, con el método keys() se obtienen
    #todos sus valores de llave = {'llave': 'valor'}.
    texto_limpio = re.findall(r'((?:\S+\s+){0,3}(?:' + '|'.join(meses.keys()) + r'))(?:\s+\S+){0,3}', preguntaUsuario,
flags=re.IGNORECASE | re.UNICODE)

    #Si la lista de tuplas que busca 3 palabras antes y después de los meses del año no está vacía, procedemos
    #a eliminar las palabras que no tienen nada que ver con una fecha, como lo son ciertas palabras clave,
    #caracteres especiales y espacios en blanco como espacios, tabuladores o saltos de línea.
    if texto_limpio:
        #re.sub(r): El método sub() se utiliza para realizar sustituciones en una cadena de texto (string)

```




```

#utilizando expresiones regulares. Este recibe 4 parámetros, la expresión regular que se quiere
#encontrar, la cadena o caracter que las sustituirá, el texto donde se buscará el patrón y modificadores
#opcionales que se pueden utilizar para personalizar el comportamiento de la búsqueda.
#En este caso se buscan palabras clave que estorben al encontrar las fechas del evento.
texto_limpio =
re.sub(r'\b(inicio|inicial|titulo|título|tituladas|titulados|fecha|hora|descripcion|descripción|final|fin|como|las|de|del
|el|y|a)\b', '', texto_limpio[0].lower(), flags=re.IGNORECASE|re.UNICODE)

#En este caso se buscan los caracteres especiales que no sean letras o números como !, ?, ., etc.
texto_limpio = re.sub(r'^\w\s', '', texto_limpio)

#En este caso se buscan espacios en blanco como tabuladores, espacios o saltos de línea.
texto_limpio = re.sub(r'\s+', ' ', texto_limpio).strip()

#split(): Método que sirve para separar las palabras en un string en función de todas las veces que
#aparezca un caracter en específico.
componentes = texto_limpio.split(" ")

mes = componentes[1].lower() #lower(): Transformar todas las letras a minúsculas.
#Si el año del evento es None o menor al actual, probablemente se haya captado de forma errónea o no
#haya sido proporcionado por el usuario, cuando esto ocurra, por default se indicará que el año es el
#actual.
#datetime.datetime.now().strftime(): Método que devuelve la fecha y hora actual en el formato
#YYYY-MM-DD HH:MM:SS, al cual se accede con el string "%Y-%m-%d %H:%M:%S".
currentYear = datetime.datetime.now().strftime("%Y")
if((componentes[2] == None) or (int(componentes[2]) < int(currentYear))):
    componentes[2] = currentYear
    #Reemplaza el contenido de la variable texto_limpio con el nuevo año de la variable componentes.
    texto_limpio = ' '.join(componentes)

#La variable mes_numerico es donde se almacena temporalmente el valor numérico correspondiente al texto
#del mes recibido en la instrucción del usuario, para que después a través del método replace() sea
#reemplazado y así se pueda convertir dicha fecha a un objeto tipo datetime.
mes_numerico = meses[mes]
texto_limpio = texto_limpio.replace(mes, mes_numerico)

#datetime.datetime.strptime(): Método de la clase datetime que toma una cadena que representa una fecha
#y hora en un formato específico y la convierte a un objeto datetime, para ello recibe las siguientes
#instrucciones en su segundo parámetro:

# - %d: Día de la fecha.
# - %m: Mes de la fecha.
# - %Y: Año de la fecha.
# - %z: Desplazamiento de la zona horaria UTC en formato ±HHMM o ±HH:MM.
#La fecha recibida usualmente indica primero el día, luego el mes y finalmente el año.
fecha_objeto = datetime.datetime.strptime(texto_limpio, "%d %m %Y")
return fecha_objeto

#Si la fecha no fue recibida en forma de texto, sino de forma numérica, se ejecuta la gestión de errores.
#MANEJO DE EXCEPCIONES: Es una parte de código que se conforma de dos partes, try y except:
# - Primero se ejecuta el código que haya dentro del try y si es que llegara a ocurrir una excepción durante
# su ejecución, el programa brinca al código del except.
# - En la parte de código donde se encuentra la palabra reservada except, se ejecuta cierta acción cuando

```



```

# ocurra el error esperado.

#Se utiliza esta arquitectura de código cuando se quiera efectuar una acción donde se espera que pueda
#ocurrir un error durante su ejecución.

try:

    print("La fecha fue proporcionada en forma de número.")

    #Esta expresión regular extrae todos los números que se encuentren entre diagonales (/).
    texto_limpio = re.findall(r"(\d+)/(\d+)/(\d+)", preguntaUsuario, flags=re.IGNORECASE | re.UNICODE)
    #La instrucción not seguida de una variable en un condicional evalúa si esta es un string o lista vacía.
    #Si no se ha podido extraer nada a través de la expresión regular que obtiene la fecha en formato
    #numérico, se crea una excepción propia llamada fechaFormatoDesconocido, la cual mostrará un mensaje en
    #consola y le hará saber al usuario que no pudo entender la fecha que dijo en su instrucción y que por
    #favor la debe repetir de otra forma.

    if not texto_limpio:
        raise fechaFormatoDesconocido("Lo siento, no pude reconocer el formato de la fecha de tu evento.")
    #Reemplaza el contenido de la variable texto_limpio con la nueva fecha en formato numérico.
    texto_limpio = ' '.join(texto_limpio[0])
    #datetime.datetime.strptime(): Método que convierte un string en cierto formato a un objeto datetime.
    fecha_objeto = datetime.datetime.strptime(texto_limpio, "%d %m %Y")
    return fecha_objeto

#La excepción propia fechaFormatoDesconocido se lanza cuando el programa no puede reconocer la fecha dicha
#indicada por el usuario, esta se declara como una clase dentro de este mismo programa.

except fechaFormatoDesconocido:
    fecha_objeto = "Lo siento, no pude reconocer la fecha de tu evento, repitemela de otra forma por favor."
    return fecha_objeto


#La hora y fecha inicial del evento se obtiene de forma separada por simplicidad y para mejorar su detección,
#pero para que esto pueda ser utilizado con la API de Google Calendar se debe mandar todo junto en una misma
#variable y en formato ISO, por dicha razón en esta función se utilizan los métodos declarados anteriormente
#para así juntar ambos datos y meterlos en una misma variable.

def __fechaHoraInicioEvento(self, preguntaUsuario):
    #__horaInicioEvento(): Función propia con modificador de acceso privado que se encarga de obtener la hora
    #inicial del evento en formato de 12 horas.
    horaInicialEvento = self.__horaInicioEvento(preguntaUsuario)

    #__fechaInicioEvento(): Función propia con modificador de acceso privado que se encarga de obtener la fecha
    #inicial del evento, no importando si esta viene indicada en forma de texto o en formato numérico.
    fechaEvento = self.__fechaInicioEvento(preguntaUsuario)

    try:
        #datetime.datetime.strptime(): Método que convierte un string en cierto formato a un objeto datetime.
        horaInicialEventoDate = datetime.datetime.strptime(horaInicialEvento, "%H:%M")
        #datetime.replace(): Cuando el método replace() se aplica a un objeto datetime, se puede utilizar para
        #cambiar la hora, minuto, segundo, etc. de su fecha.
        fechaHora_Inicial = fechaEvento.replace(hour = horaInicialEventoDate.hour, minute = horaInicialEventoDate.minute)
        #datetime.isoformat(): El método isoformat() convierte el formato del objeto datetime a ISO.
        fechaHoraInicial_ISO = fechaHora_Inicial.isoformat()

        return fechaHoraInicial_ISO

```



```

#Si no se pudo identificar el formato de la fecha se lanza la excepción TypeError y se proporciona un
#mensaje al usuario.
except TypeError:
    fechaHoraInicial_ISO = "Lo siento, no pude reconocer la fecha y hora de tu evento, repitemela de otra forma por
favor."

    return fechaHoraInicial_ISO

#Una vez que se ha recabado la fecha y hora inicial del evento se analiza si existe alguna otra hora dentro de
#la instrucción que se encuentre después de ciertas palabras clave y antes de las palabras am o pm, para que
#así se considere la fecha de término del evento dada por el usuario, pero si esta hora no fue dada, por
#defecto se declarará que la hora de término sea 1 hora después, ya que la API de Google Calendar lo requiere.
def __fechaHoraFinalEvento(self, preguntaUsuario):
    preguntaUsuario = preguntaUsuario.lower() #lower(): Transformar todas las letras a minúsculas.
    instruccionHora = preguntaUsuario.strip() #lower(): Elimina los caracteres en blanco del string.
    #Esta expresión regular extrae todos los números que se encuentren entre ciertas palabras clave y las
    #palabras am o pm, pero como se utiliza la instrucción ?:, las palabras am o pm no serán guardadas en una
    #tupla dentro de la lista extraída ni tampoco las palabras clave, solo serán extraídos y separados los
    #números.
    fechaHoraFinal = re.findall(r"\b(?:a|termine|hasta\s+las?)\s+(\d+):(\d+) (?:am|pm)?\b", instruccionHora,
re.IGNORECASE|re.UNICODE)

    #Condicional que se ejecuta si la hora de finalización del evento fue devuelta por la expresión regular.
    if fechaHoraFinal:
        #join(): Este método toma una lista o tupla de strings como argumento y los concatena, juntando cada
        #string con el carácter especificado.
        horaFinalEvento = ":".join(fechaHoraFinal[0])

        #Para que el string que representa la hora pueda ser convertido a un objeto datetime con formato de 12
        #horas, se debe indicar si la hora de este es pm o am, pero para que esto sea entendido por el método
        #strptime(), se debe indicar en mayúsculas.
        if("pm" in preguntaUsuario):
            horaFinalEvento += " PM"
        else:
            horaFinalEvento += " AM"

        #datetime.datetime.strptime(): Método de la clase datetime que toma una cadena que representa una fecha
        #y hora en un formato específico y la convierte a un objeto datetime, para ello recibe las siguientes
        #instrucciones en su segundo parámetro:
        # - %H: Hora en formato de 24 horas (00-23).
        # - %I: Hora en formato de 12 horas (01-12). Para esto se debe indicar si la hora es AM o PM.
        # - %p: AM o PM (funciona con %I para indicar si el formato de la hora es en la tarde o la mañana).
        # - %M: Minutos (00-59).
        # - %S: Segundos (00-59).
        # - %f: Microsegundos (000000-999999).
        # - %z: Desplazamiento de la zona horaria UTC en formato ±HHMM o ±HH:MM.
        horaFinalEvento = datetime.datetime.strptime(horaFinalEvento, "%I:%M %p") #Hora en formato de 12 horas.
        #Conversión de objeto datetime con formato de 12 horas en formato de 24 horas. Esto se hace para que sea
        #compatible con la hora actual retornada por medio del método datetime.datetime.now().strftime("%H:%M").

```



```

        horaFinalEvento = horaFinalEvento.strftime("%H:%M")

        #Una vez extraída la hora de finalización, se declara la fecha de finalización, que será la misma a la
        #inicial.

        fechaEvento = self.__fechaInicioEvento(preguntaUsuario)

        #Después se intentará cambiar el formato de la hora final e incluirla con la fecha final.
    try:
        horaFinalEventoDate = datetime.datetime.strptime(horaFinalEvento, "%H:%M")
        fechaHora_Final = fechaEvento.replace(hour = horaFinalEventoDate.hour, minute = horaFinalEventoDate.minute)
        fechaHoraFinal_ISO = fechaHora_Final.isoformat()

        return fechaHoraFinal_ISO

    #Pero si no se logra realizar esta conversión es porque la fecha final no fue dada por el usuario, por lo
    #cual se establecerá por default 1 hora después de la dada por la fecha y hora inicial.
    except UnboundLocalError:
        print("No existe una hora final del evento, pero por default se establecerá 1 hora después.")
        horaInicialEvento = self.__horaInicioEvento(preguntaUsuario)
        fechaEvento = self.__fechaInicioEvento(preguntaUsuario)

        try:
            horaInicialEventoDate = datetime.datetime.strptime(horaInicialEvento, "%H:%M")

            #datetime.timedelta(): Método que se utiliza para representar una duración o un intervalo de tiempo
            #transcurrido, el cual mayormente es usado para realizar operaciones matemáticas con fechas y horas,
            #como sumar o restar intervalos de tiempo a objetos datetime.

            horaFinalEventoDate = horaInicialEventoDate + datetime.timedelta(hours = 1)
            fechaHora_Final = fechaEvento.replace(hour = horaFinalEventoDate.hour, minute = horaFinalEventoDate.minute)
            fechaHoraFinal_ISO = fechaHora_Final.isoformat()

            return fechaHoraFinal_ISO

        #Si no se reconoce la hora aún así, se maneja la excepción TypeError lanzada por el error.
        except TypeError:
            print("La fecha fue proporcionada en un formato desconocido.")
            fechaHoraFinal_ISO = "Lo siento, no pude reconocer la fecha y hora de tu evento, repítemela de otra forma por
            favor."

            return fechaHoraFinal_ISO

#Para definir un evento de Google Calendar el key start y end del diccionario body perteneciente al método
#events().create().insert() necesita recibir la fecha y hora en formato ISO, el cual es extraído a través de
#la librería datetime.
def crearEvento(self, preguntaUsuario):
    respuestaEventoCalendario = ""

    try:
        tituloEvento = self.__tituloEvento(preguntaUsuario)
        time.sleep(0.5)

        descripcionEvento = self.__descripcionEvento(preguntaUsuario)
        time.sleep(0.5)

        horaFechaInicioEvento = self.__fechaHoraInicioEvento(preguntaUsuario)
        time.sleep(0.5)

        horaFechaFinalEvento = self.__fechaHoraFinalEvento(preguntaUsuario)

```



```

time.sleep(0.5)

#servicioAPIGoogleCalendar(): Función propia de la clase googleCalendar_auth que sirve para establecer una
#conexión con la API de Google Calendar.
calendar_service = self.servicioAPI

#Con la variable que utilizó el método servicioAPIGoogleCalendar() se ejecuta el método events().insert()
#que recibe dos parámetros, el primero calendarId indica a qué calendario nos estamos refiriendo ya que
#pueden existir varios dentro de Google Calendar, aunque casi siempre se utiliza el principal, denotado por
#el valor de primary, en el segundo parámetro llamado body se indican las características del evento como
#su título, descripción, fecha y hora de inicio, zona horaria y fecha y hora de finalización.
eventoGoogleCalendar = calendar_service.events().insert(
    calendarId = "primary",
    body = {
        "summary": tituloEvento,
        "description": descripcionEvento,
        #timeZone: La zona horaria de cada locación se obtiene del siguiente enlace al buscar tz database
        #names: https://en.wikipedia.org/wiki/List_of_tz_database_time_zones
        "start": {"dateTime": horaFechaInicioEvento, "timeZone": "America/Mexico_City"},
        "end": {"dateTime": horaFechaFinalEvento, "timeZone": "America/Mexico_City"},
    }
).execute()

#Impresión en consola del resultado del evento agendado.
print("Evento creado con éxito!!")

print("Título:\t\t\t" + str(eventoGoogleCalendar["summary"]))
print("Descripción:\t\t\t" + str(eventoGoogleCalendar["description"]))
print("Fecha y Hora de Inicio:\t" + str(eventoGoogleCalendar["start"]))
print("Fecha y Hora de Fin:\t" + str(eventoGoogleCalendar["end"]))

locale.setlocale(locale.LC_TIME, 'es_ES.UTF-8')
horaFechaInicioEvento = datetime.datetime.fromisoformat(horaFechaInicioEvento)
horaFechaFinalEvento = datetime.datetime.fromisoformat(horaFechaFinalEvento)
fechaFormateadaInicio = horaFechaInicioEvento.strftime("%d de %B de %Y")
fechaFormateadaFinal = horaFechaFinalEvento.strftime("%d de %B de %Y")

respuestaEventoCalendario = f"""Evento creado con éxito!! con título de: {tituloEvento}, su descripción es:
{descripcionEvento}, su hora y fecha de inicio es el {fechaFormateadaInicio} y
finaliza el {fechaFormateadaFinal}."""

respuestaEventoCalendario = re.sub(r'\s+', ' ', respuestaEventoCalendario).strip()

#webbrowser.open(): Método que abre un navegador web en la dirección URL especificada.
webbrowser.open("https://calendar.google.com/")

except:
    respuestaEventoCalendario = """Lo siento, no entendí bien las características del evento que quieres agendar,
recuerda que para crear un evento en Google Calendar debes indicar su título,
descripción, fecha y hora de inicio y fecha y hora de finalización."""

    respuestaEventoCalendario = re.sub(r'\s+', ' ', respuestaEventoCalendario).strip()

return respuestaEventoCalendario

```

#Creación de una excepción propia que sea lanzada cuando el formato de la fecha dada hacia el calendario se



```
#proporcione en forma numérica.
class fechaFormatoDesconocido(Exception):
    def __init__(self, mensaje):
        super().__init__(mensaje)
```

Clase: googleCalendar_auth

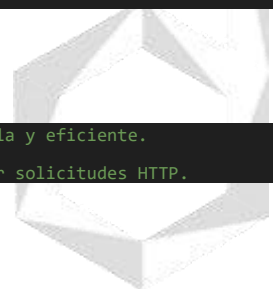
```
#IMPORTACIÓN DE LIBRERÍAS:
import os          #os: Librería que permite acceder a funciones y métodos relacionados con el sistema operativo.
#Serialización es el proceso de convertir un objeto Python en una secuencia de bytes, mientras que la
#deserialización es el proceso de reconstruir un objeto Python a partir de una secuencia de bytes. En otras
#palabras, la librería pickle permite guardar y cargar objetos Python de manera eficiente.
import pickle      #pickle: Librería que se utiliza para serializar y deserializar objetos Python.
from googleapiclient.discovery import build          #googleapiclient: Permite utilizar la Google API.
from google_auth_oauthlib.flow import InstalledAppFlow #google_auth_oauthlib: Maneja el flujo de autenticación.
from google.auth.transport.requests import Request   #Request: Gestiona solicitudes HTTP de Google.

SCOPES = ["https://www.googleapis.com/auth/calendar"]
CREDENCIALS_FILE = "C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/API_Keys/credencialesGoogleCalendar.json"

#servicioAPIGoogleCalendar(): Función propia que realiza la autenticación de la Google API, esto no es tan
#importante entenderlo línea a línea ya que solamente sirve para poder configurar la utilización de la
#API de Google Calendar.
def servicioAPIGoogleCalendar():
    creds = None
    if os.path.exists("token.pickle"):
        with open("token.pickle", "rb") as token:
            creds = pickle.load(token)
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file(
                CREDENCIALS_FILE, SCOPES
            )
            creds = flow.run_local_server(port = 0)
        with open("token.pickle", "wb") as token:
            pickle.dump(creds, token)
    service = build("calendar", "v3", credentials = creds)
    return service
```

Clase: mapaAsistente

```
import requests #requests: Biblioteca que sirve para realizar solicitudes HTTP de manera sencilla y eficiente.
import urllib   #urllib: Librería que proporciona herramientas para trabajar con URLs y realizar solicitudes HTTP.
```



```

from PIL import Image    #PIL: La librería Pillow sirve para mostrar una imagen con python extraída de una URL.
import io                #io: Librería que permite la manipulación de los archivos y carpetas de nuestro ordenador.
import re                #re: Librería que permite utilizar expresiones regulares para detectar patrones.

class widgetDireccionesMapa:

    #CONSTRUCTOR O INICIALIZADOR DE LA CLASE: En él se declaran los parámetros que recibe la clase, que además se
    #utilizarán en los demás métodos, estos a fuerza deben tener un valor.

    #self: La instrucción self se utiliza para hacer referencia al objeto que se está manipulando cuando se instancia
    #la clase. Por eso es que a través de la misma nomenclatura de un punto se accede a los distintos atributos y/o
    #métodos con un objeto desde fuera de la clase.

    def __init__(self, parametro_de_la_clase):

        #ChatGPT API key

        self.MapQuestApiKey = parametro_de_la_clase

    def direccionesMapa(self, peticionDireccion):

        respuestaMapa = ""
        punto_inicial = ""
        punto_final = ""

        #re.findall(r): El método findall() utilizado para encontrar todas las ocurrencias de una expresión regular en
        #una cadena de texto (string) recibe 3 parámetros, la expresión regular que se quiere encontrar, la cadena de
        #caracteres donde se buscará y modificadores opcionales que se pueden utilizar para personalizar el
        #comportamiento de la búsqueda.

        # - pattern: Parámetro que indica la expresión regular utilizada para extraer partes específicas del string.
        # Las expresiones regulares más comunes son:
        # - Búsqueda de números en un string:
        #     - \d: Expresión regular que busca un dígito del 0 al 9 en un string.
        #     - \d+: Expresión regular que coincide con uno o más dígitos del 0 al 9.
        #     - \d{2,4}: Expresión regular que coincide con una secuencia de dígitos de 2 al 4 específicamente,
        #         osea 234.
        # - Búsqueda de letras en un string:
        #     - \w: Expresión regular que busca una letra mayúscula o minúscula, un dígito del 0 al 9 o un guión
        #         bajo.
        #     - \w+: Expresión regular que coincide con uno o más caracteres de los descritos anteriormente.
        #     - [A-Za-z]: Expresión regular que busca solo una letra mayúscula o minúscula.
        #     - \b: Expresión regular que representa el inicio o final de una palabra en un string, usualmente se
        #         incluyen dos para indicar que se debe buscar un patrón en cada palabra. Por ejemplo, si se declara la
        #         expresión regular \bHola\b, esto extraerá todas las palabras Hola dentro de un string.
        # - Búsqueda de palabras en un string (grupos de captura):
        #     - (patrón1|patrón2): Expresión regular que crea un grupo de captura para identificar uno o varios
        #         patrones específicos.
        #     - |: Compuerta OR utilizada para reconocer uno o varios patrones dentro de un grupo de captura.
        # - Búsqueda de espacios en blanco, puntos y paréntesis en un string:
        #     - \s: Expresión regular que coincide con un carácter de espacio vacío, como un espacio, tabulador (\t),
        #         salto de línea (\n), etc.
        #     - \.: Expresión regular que encuentra un punto en un string.

```




```

# - \(: Expresión regular que encuentra un paréntesis de apertura en un string.
# - \): Expresión regular que encuentra un paréntesis de cierre en un string.
# - Búsqueda de repeticiones en un string:
# - *: Encuentra cero o más repeticiones de la expresión regular que tenga a la izquierda.
# - +: Encuentra una o más repeticiones de la expresión regular que tenga a la izquierda.
# - ?: Encuentra cero o una repetición de la expresión regular que tenga a la izquierda. Esto significa
#     que la expresión regular que la precede es opcional y puede aparecer una vez o no aparecer en
#     absoluto en la cadena que se está buscando.
# - {2,4}: Coincide con 2, 3 o 4 repeticiones de la expresión regular que tenga a la izquierda.
# - Anclaje de patrones:
# - ^: Cuando se coloca al principio de un patrón indica que la coincidencia debe encontrarse al comienzo
#     de la línea de texto. Por ejemplo, si se declara la expresión regular ^Hola, esto será cierto solo
#     cuando la palabra Hola se encuentre al inicio del string.
# - $: Cuando se coloca al final de un patrón indica que la coincidencia debe encontrarse al final de la
#     línea de texto. Por ejemplo, si se declara la expresión regular $mundo, esto será cierto solo cuando
#     la palabra mundo se encuentre al final del string.
# - ?: : Cuando se coloca un signo de interrogación seguido de dos puntos se está indicando que es un
#     grupo no captador, esto significa que se tomará en cuenta para hacer coincidencias, pero no será
#     incluido dentro de la lista de tuplas.
# - string: Palabra en la que se desea buscar patrones a través de la expresión regular especificada.
# - flags: Modificadores opcionales que personalizan el comportamiento de la búsqueda.
# - re.IGNORECASE: Realiza la búsqueda sin distinción entre mayúsculas y minúsculas.
# - re.MULTILINE: Permite que el patrón coincida con múltiples líneas en la cadena.
# - re.DOTALL: Hace que el carácter . en el patrón coincida también con el carácter de nueva línea \n.
#Esta expresión regular busca números en un string.
patronDirecciones = re.findall(r"\b(desde|entre|para|inicio|a|de|ir)\b\s+(.*?)\s+\b(a|y|hasta|final)\b\s+(.*?)$",
peticionDireccion, re.IGNORECASE|re.UNICODE)

#Al utilizar el método re.findall() se retorna una lista de tuplas de todas las ocurrencias que encuentre,
#para recorrer estos y filtrarlos se usa el método propio filtroTexto(), que será declarado posteriormente
#y se encargará de eliminar palabras clave que estorben al obtener el punto de inicio y final al buscar
#los puntos iniciales y finales de la ruta.
if patronDirecciones:
    for coincidencia in patronDirecciones:
        #strip(): Método utilizado para eliminar los caracteres especificados en una cadena, como espacios
        #saltos de línea (\n), tabuladores (\t), etc. Si no se le pasa ningún parámetro, eliminará los
        #espacios del string.
        punto_inicial = self.__filtroTexto(coincidencia[1].strip())
        punto_final = self.__filtroTexto(coincidencia[3].strip())
    else:
        print("No se encontró un patrón de inicio y destino en la solicitud.")

print("Punto Inicial: " + punto_inicial)
print("Punto Final: " + punto_final + "\n")

```



```

#El endpoint de una API se refiere a una URL específica a través de la cual se pueden realizar peticiones, en
#el caso de la API MapQuest, esta se obtiene al ingresar a la opción de Documentation -> APIs -> Directions
#API -> Route -> GET -> Route -> Resource URL y al final de esa URL agregar un signo de interrogación. Este
#endpoint sirve para obtener la distancia entre dos puntos, el tiempo del recorrido en auto, consumo de
#combustible, las indicaciones para llegar desde un punto al otro, etc.
MapQuestEndpoint = "https://www.mapquestapi.com/directions/v2/route?"

#urllib.parse.urlencode(): Método que convierte un objeto de diccionario o una secuencia de tuplas de dos
#elementos en una cadena de consulta URL, para ello anteriormente se tuvo que haber declarado la URL a la que
#se realizará la consulta y al final haber puesto un signo de interrogación.
peticionAPI = MapQuestEndpoint + urllib.parse.urlencode({
    "key": self.MapQuestApiKey,
    "from": punto_inicial,
    "to": punto_final
})

print("La consulta realizada a la herramienta MapQuest Developer para obtener los datos de la ruta fue:\n" + peticionAPI)

#requests.get(endpoint).json(): A través de la librería request se puede usar cualquier método HTTP, en este
#caso se utiliza el método get que devuelve la respuesta retornada por una API en una endpoint específica,
#una vez que se recibe la respuesta del servidor, se utiliza el método .json() para interpretar la respuesta
#como dato tipo JSON o en forma de diccionario, que es su equivalente pero en Python.
respuestaAPI = requests.get(peticionAPI).json()

#En el JSON de la respuesta de la API si su key statuscode tiene un value de 0, esto significa que se han
#recibido correctamente sus datos.
if (respuestaAPI["info"]["statusCode"] == 0):
    print("Respuesta recibida correctamente de la API de MapQuest.")
    respuestaMapa += " la información de la ruta que va desde " + punto_inicial + " hasta " + punto_final + " es la
siguiente... "

    duracionRuta = respuestaAPI["route"]["formattedTime"] #Tiempo del recorrido.
    respuestaMapa += "La duración del viaje es de " + duracionRuta + " aproximadamente y "
    distanciaRuta = respuestaAPI["route"]["distance"] * 1.60934 #La distancia en millas se convierte a km.
    #"{:.2f}".format(): Con esta instrucción se indica que un número decimal solo muestre dos decimales.
    respuestaMapa += "La distancia recorrida en auto durante el viaje es de " + str("{:.2f}".format(distanciaRuta))
+ " kilometros. "
else:
    print("Respuesta recibida erróneamente de la API de MapQuest.")
    respuestaMapa += "Lo siento di0, no pude procesar bien la ruta, podrías repetirmela por favor? "

#El endpoint de una API se refiere a una URL específica a través de la cual se pueden realizar distintas
#peticiones, en este caso se utiliza para obtener el mapa que muestra la ruta entre dos puntos, para ello
#ingresaremos a la opción de Documentation -> APIs -> Static Map API -> GET -> Map -> Resource URL -> Adding
#a Route to the Map y al final de esa URL agregar de nuevo un signo de interrogación.
MapQuestMapEndpoint = "https://www.mapquestapi.com/staticmap/v5/map?"

#urllib.parse.urlencode(): Método que convierte un objeto de diccionario o una secuencia de tuplas de dos
#elementos en una cadena de consulta URL, para ello anteriormente se tuvo que haber declarado la URL a la que
#se realizará la consulta y al final haber puesto un signo de interrogación.
peticionMapaAPI = MapQuestMapEndpoint + urllib.parse.urlencode({

```



```

        "key": self.MapQuestApiKey,
        "start": punto_inicial,
        "end": punto_final,
        "size": "700,700"
    })

    print("La consulta realizada a la herramienta MapQuest Developer para obtener el mapa de la ruta fue:\n" + peticionMapaAPI
+ "\n")

    respuestaMapa += "He desplegado una imagen con el mapa de la ruta, en esta el círculo verde marca el punto de inicio
y el rojo el final."

    print("Hola di_cer0." + respuestaMapa + "\n")

    #requests.get(endpoint): A través de la librería request se puede usar cualquier método HTTP, en este caso se
    #utiliza el método get, que devuelve la respuesta retornada por una API en una endpoint específica.
    imagenMapaAPI = requests.get(peticionMapaAPI)

    #io.BytesIO(): Método que sirve para crear un flujo de bytes en memoria que puede utilizarse para almacenar
    #datos binarios, como imágenes, audio y video. Este se almacena en un archivo temporal perteneciente a la
    #carpeta %temp% = C:\Users\diego\AppData\Local\Temp.
    imagen_bytes = io.BytesIO(imagenMapaAPI.content)

    #PIL.Image.open(): Método que se utiliza para abrir una imagen desde un archivo local o desde una fuente de
    #datos en memoria, como bytes. Después se aplica el método show() para abrir dicha imagen.
    Image.open(imagen_bytes).show()

    return respuestaMapa

def __filtroTexto(self, texto):
    #re.sub(r): El método sub() se utiliza para realizar sustituciones en una cadena de texto (string) utilizando
    #expresiones regulares. Este recibe 4 parámetros, la expresión regular que se quiere encontrar, la cadena o
    #carácter que las sustituirá, el texto original donde se buscará el patrón y modificadores opcionales que se
    #pueden utilizar para personalizar el comportamiento de la búsqueda.
    #En este caso se buscan palabras clave que estorben al encontrar las direcciones de inicio y final dadas al
    #asistente virtual.
    texto_limpio =
re.sub(r'\b(desde|entre|llegar|muestrame|distancia|muéstrame|mismo|para|inicio|ruta|rapida|rápida|lenta|cual|cuál|mas|más
|una|de|ir|es|la|a)\b', '', texto, flags=re.IGNORECASE|re.UNICODE)

    #Posteriormente se eliminan todos los caracteres especiales que no sean letras, números o espacios.
    texto_limpio = re.sub(r'^\w\s', '', texto_limpio)

    return texto_limpio

```

Clase: visionArtificialAsistente: Reconocimiento de Colores en Tiempo Real

```

#IMPORTACIÓN DE LIBRERÍAS:
import cv2 #Librería OpenCV: Sirve para todo tipo de operaciones de visión artificial
import numpy as np #Librería numpy: Realiza operaciones matemáticas complejas

class visionArtificial():
    # _dibujarContorno(): Esta función propia se encargará de identificar los colores de las figuras que aparezcan

```

```

#en la cámara para posteriormente rodearlas con un contorno del color BGR (RGB volteado) indicado en su segundo
#parámetro.

def __dibujarContorno(self, mascaraImg, color_contorno, frameContorno):
    #cv2.findContours(): Este método sirve para encontrar los contornos en una imagen, a este se le pasa como
    #primer parámetro la imagen en escala de grises, después cabe mencionar que en las imágenes existen varios
    #tipos de contornos y estos pueden tener jerarquías de dentro hacia fuera o pueden no tenerlas, esto depende
    #del segundo parámetro que se le pase al método, en este caso vamos a utilizar el parámetro cv2.RETR_EXTERNAL
    #para que muestre solo los bordes externos que encuentre, luego se le debe indicar en el tercer parámetro la
    #forma en la que se va a guardar la información recabada, ya que puede guardar todos los puntos del contorno
    #identificado o solo guardar los puntos de las esquinas que conforman el contorno, para que guarde todos los
    #puntos de los contornos se usa el atributo cv2.CHAIN_APPROX_NONE, pero si se busca guardar solo sus esquinas
    #se usa el atributo cv2.CHAIN_APPROX_SIMPLE.
    contornos, _ = cv2.findContours(mascaraImg, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    #Bucle for que recorre todas las líneas de los contornos para calcular el área del objeto.
    for lineas in contornos:
        #cv2.contourArea(): Método que permite definir el área de una figura en función de su contorno.
        area = cv2.contourArea(lineas)
        #Con este condicional se filtran las figuras pequeñas que se pueden detectar, para que solo detecte las
        #grandes.
        if(area > 1500):
            #cv2.convexHull(): Método que filtra un poco el borde de la imagen para reducir su ruido.
            contornoFigGrande = cv2.convexHull(lineas)
            #cv2.drawContours(): Método que sirve para dibujar los contornos recabados con el método
            #cv2.findContours(), para ello primero se debe indicar en qué imagen o frame se va a dibujar el
            #contorno, no debe ser el mismo de donde se obtuvieron los contornos, después se indica en el
            #segundo parámetro la variable en donde están almacenados los contornos, en el tercer parámetro se
            #establece si se quiere que el contorno sea hueco o sólido, como se busca que sea sólido se pone -1,
            #si fuera hueco, en esta parte se indicarían los píxeles del perímetro (borde) del contorno,
            #posteriormente se indica el color y finalmente los contornos que quiero que muestre.
            cv2.drawContours(frameContorno, [contornoFigGrande], 0, color_contorno, 3)
            #cv.moments(): Método que permite obtener el centroide de la figura rodeada por un contorno.
            momentoCentroide = cv2.moments(lineas)
            if (momentoCentroide["m00"] == 0):
                momentoCentroide["m00"] == 1
            centroide_x = int(momentoCentroide["m10"]/momentoCentroide["m00"])
            centroide_y = int(momentoCentroide["m01"]/momentoCentroide["m00"])
            tipoLetra = cv2.FONT_HERSHEY_COMPLEX #Tipo de letra sans-serif del texto que indica el color.
            if (color_contorno == [0, 255, 255]):
                #cv2.putText(): Método utilizado para agregar texto a una imagen o un fotograma de video, para
                #ello recibe como parámetros la imagen donde se quiere colocar, el texto, las coordenadas donde
                #se posicionará, el tipo de letra, la escala de su tamaño, su color, grosor, y tipo de línea para
                #dibujar las letras, que usualmente es el cv2.LINE_AA.
                cv2.putText(frameContorno, "Amarillo", (centroide_x + 10, centroide_y), tipoLetra, 0.75, (0, 255, 255),
1, cv2.LINE_AA)
            elif (color_contorno == [0, 0, 255]):

```



```

        cv2.putText(frameContorno, "Rojo", (centroide_x + 10, centroide_y), tipoLetra, 0.75, (0, 0, 255), 1,
cv2.LINE_AA)

        elif (color_contorno == [255, 0, 0]):
            cv2.putText(frameContorno, "Azul", (centroide_x + 10, centroide_y), tipoLetra, 0.75, (0, 0, 255), 1,
cv2.LINE_AA)

        elif (color_contorno == [0, 255, 0]):
            cv2.putText(frameContorno, "Verde", (centroide_x + 10, centroide_y), tipoLetra, 0.75, (0, 0, 255), 1,
cv2.LINE_AA)

#reconocerColoresVideo(): Esta función utiliza el método __dibujarContorno() para que por medio de ciertas
#máscaras que identifican ciertos colores en el video con formato HSV se rodee las figuras que detecten dicho
#color en la cámara.
def reconocerColoresVideo(self):
    #DECLARACIÓN DE COLORES CON GAMA HSV:
    #La mejor manera de detectar los colores en una imagen o video es a través del código de color HSV (Hue,
    #Saturation y View), ya que este no solo considera el color sino su intensidad, luz, etc.
    # - Hue = Matiz: Representa todos los tonos de colores y abarca valores de 0 a 180.
    # - Saturation = Saturación: Representa la intensidad del color y abarca valores de 0 a 255.
    # - View = Valor: Representa la luz y abarca valores también de 0 a 255.
    #En la gráfica incluida en el documento 9.-Asistente Virtual - SpeechRecognition, Whisper y LangChain se
    #muestra los colores que se pueden abarcar variando el Matiz (Hue) corresponde al eje x y la Saturación
    #(Saturation) correspondiente al eje y, el cual aumenta su valor de arriba hacia abajo.
    amarillo_Obscuro = np.array([20, 190, 20], np.uint8)
    amarillo_Claro = np.array([30, 255, 255], np.uint8)

    rojo_Obscuro_Izq = np.array([0, 100, 20], np.uint8)
    rojo_Claro_Izq = np.array([5, 255, 255], np.uint8)
    rojo_Obscuro_Der = np.array([175, 100, 20], np.uint8)
    rojo_Claro_Der = np.array([180, 255, 255], np.uint8)

    azul_Obscuro = np.array([85, 200, 20], np.uint8)
    azul_Claro = np.array([125, 255, 255], np.uint8)

    verde_Obscuro = np.array([45, 100, 20], np.uint8)
    verde_Claro = np.array([75, 255, 255], np.uint8)

    #cv2.VideoCapture(): Método de python para acceder a la webcam, como parámetro solo se enlista el número de
    #webcams a las que se quiere acceder, empezando a contar desde el índice 0, 1, 2, etc. O se le puede pasar
    #entre comillas la ruta de un video para que lo reproduzca.
    camara = cv2.VideoCapture(0)

    #.isOpened(): Método que se aplica a un objeto de OpenCV que haya abierto una webcam, este identifica si la
    #ventana de la webcam está abierta o no, para que se realice el análisis de la imagen en el video.
    if not camara.isOpened():

```



```

#Mensaje que se imprimirá en consola cuando no se pueda abrir la webcam del ordenador.
print("No es posible abrir la cámara")

exit()

#Bucle while: Cuando se analice un video, se debe realizar dentro de un bucle while para que se ejecute
#hasta que la ventana donde se muestra el video se cierre.
while True:

    #.read(): Método que se aplica a un objeto de OpenCV que haya abierto una webcam, este método obtiene
    #dos resultados:

    # - ret: Variable booleana que puede ser true si hay un video (fotograma) capturado y false si no hay
    # fotograma que leer, este se usa más que nada para crear un if cuando se quiere leer un video ya
    # existente en vez de grabar y mostrar uno nuevo o si se quiere asegurar que la webcam está grabando.
    # - frame: La variable frame devuelve el video capturado.
    ret, frame = camara.read()

    #Condicional que se ejecuta cuando no se pudo obtener un fotograma.
    if not ret:
        print("No es posible obtener la imagen")
        break
    else:
        #OBTENCIÓN DE COLORES HSV:
        #cvtColor(): Método de la librería OpenCV que recibe como primer parámetro una imagen RGB y en su
        #segundo parámetro recibe un atributo de OpenCV llamado cv2.COLOR_BGR2HSV para convertir una imagen
        #RGB a su formato de colores HSV.
        frameHSV = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        #cv2.inRange(): Este método sirve para umbralizar una imagen hsv, en ella se le pasa como primer
        #parámetro la imagen en colores hsv que se va a umbralizar, a continuación se indica el valor mínimo
        #y máximo que puede alcanzar el umbral.
        mascaraAmarilla = cv2.inRange(frameHSV, amarillo_Obscuro, amarillo_Claro)

        mascaraRojaIzq = cv2.inRange(frameHSV, rojo_Obscuro_Izq, rojo_Claro_Izq)
        mascaraRojaDer = cv2.inRange(frameHSV, rojo_Obscuro_Der, rojo_Claro_Der)
        mascaraRoja = cv2.add(mascaraRojaIzq, mascaraRojaDer)

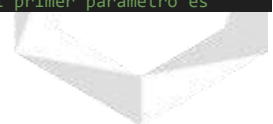
        mascaraAzul = cv2.inRange(frameHSV, azul_Obscuro, azul_Claro)

        mascaraVerde = cv2.inRange(frameHSV, verde_Obscuro, verde_Claro)

        #__dibujarContorno(): Esta función propia recibe dos parámetros, el primero es la máscara del color
        #que identificará y el segundo es un array que representa un color BGR para que pinte el contorno de
        #ese color, BGR es lo mismo que RGB pero al revés.
        self.__dibujarContorno(mascaraAmarilla, [0, 255, 255], frame)
        self.__dibujarContorno(mascaraRoja, [0, 0, 255], frame)
        self.__dibujarContorno(mascaraAzul, [255, 0, 0], frame)
        self.__dibujarContorno(mascaraVerde, [0, 255, 0], frame)

        #imshow(): Con el método se puede mostrar una ventana con una imagen o video, el primer parámetro es

```



```

#el nombre de la ventana donde aparecerá la imagen o video y el segundo parámetro es la imagen
#recopilada con el método cv2.imread() o el video recopilado con el método .read() aplicado a un
#video con el método cv2.VideoCapture().
cv2.imshow('Reconocimiento colores', frame)

#waitKey(): Método que permite a los usuarios mostrar una ventana durante un número de milisegundos
#determinados si su parámetro es un número mayor a 1 o hasta que se presione cualquier tecla. Toma
#tiempo en milisegundos como parámetro y espera el tiempo dado para destruir la ventana:
# - Si se pasa 0 como argumento, espera hasta que se presiona cualquier tecla.
# - Si se pasa 1 como argumento, espera hasta que se presione una tecla específica para cerrar la
#   ventana.
#El método waitKey() está monitoreando si se presiona una tecla o no, este modo se activa si se le
#pasa como parámetro un número 1, de esta manera:
# - Si no se presiona una tecla retorna un valor -1
# - Si se presiona una tecla, retorna un valor ASCII correspondiente a la tecla que se oprimió, con
#   el método ord() podemos indicar el código ASCII de la letra del teclado que indiquemos en su
#   parámetro.
if cv2.waitKey(1) == ord('c'): #Se cierra el programa cuando se presione la letra q
    #cv2.VideoCapture().release(): Con este método se libera la webcam, tanto en software como en
    #hardware.
    camara.release()
    #destroyAllWindows(): Función que permite a los usuarios destruir todas las ventanas en
    #cualquier momento. No toma ningún parámetro y no devuelve nada, esto se incluye para que al
    #cerrar la ventana después del método waitKey() se destruyan para poder utilizarse en otra cosa.
    cv2.destroyAllWindows()
    break

```

Clase: reconocimientoFacialAsistente: Entrenamiento con Machine Learning

```

#IMPORTACIÓN DE LIBRERÍAS:
import cv2          #Librería OpenCV: Sirve para todo tipo de operaciones de visión artificial
import os           #os: Librería que permite acceder a métodos relacionados con el sistema operativo (so).
#threading: Librería que permite crear y gestionar los hilos (threads) de un programa. Los hilos son subprocesos
#que le permiten a un programa realizar múltiples tareas al mismo tiempo, esto se añade debido a la clase de
#visión artificial, ya que si esto no se añade, el asistente virtual detendrá su ejecución mientras se ejecuta
#la clase visionArtificialAsistente.
import threading

import pygame       #pygame: librería de multimedia que permite mostrar gráficos, reproducir sonido, etc.
import subprocess   #subprocess: Librería que se usa para interactuar con el so, ejecutar programas externos, etc.
import winsound     #winsound: Biblioteca para reproducir sonidos utilizando el sistema operativo Windows.

#pygame.mixer.init(): El método init() inicializa el objeto mixer de la librería pygame que sirve para reproducir
#sonidos con Python.
pygame.mixer.init()

```




```

rutaDatosFaciales = "C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/UsuariosReconocimientoFacial"

#os.listdir(): Método que extrae todos los nombres de los archivos y carpetas contenidos en un directorio.
dataImágenesUsuarios = os.listdir(rutaDatosFaciales)

#cv2.face.LBPHFaceRecognizer_create(): Este método crea un objeto de reconocimiento de rostros utilizando el
#algoritmo de reconocimiento de patrones basado en histogramas locales de patrones binarios (Local Binary
#Patterns Histograms, LBPH) de OpenCV que se utiliza para reconocer caras en imágenes y luego crear un histograma
#de patrones binarios, el cual será utilizado para identificar cada rostro una vez que haya sido entrenado con un
#conjunto de imágenes etiquetadas de una cara en específico.
reconocimientoFacial = cv2.face.LBPHFaceRecognizer_create()
#cv2.face.LBPHFaceRecognizer_create().read(): El método read() se utiliza para leer un modelo preentrenado
#con datos faciales, para así reconocer cierto rostro. Los datos del modelo se obtuvieron al analizar un
#video del usuario a través del archivo 1_DatosFaciales.py y se entrenó con el programa 2_EntrenamientoFacial.py
reconocimientoFacial.read("C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/ReconocimientoFacial_AsistenteVirtual_MarkI/modeloReconocimientoFacial.xml")

#cv.CascadeClassifier(): Este es un clasificador en cascada, osea que dependiendo de ciertas características del
#reconocimiento facial, puede ir segmentando el video o imagen hasta identificar un rostro, ya sea de hombre,
#mujer, perro, gato, etc. porque cada uno tiene características especiales, para que funcione esto, se debe
#indicar como parámetro del método la base de datos descargada de la documentación de Github de OpenCV con
#extensión xml.
#https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
#Para la identificación de rostros, específicamente se utiliza la base de datos llamada
#haarcascade_frontalface_default.xml:
#https://github.com/opencv/opencv/tree/4.x/data/haarcascades
rutaDatosEntrenamientoFacial = "C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/API_Keys/haarcascade_frontalface_default.xml"
clasificacionFacial = cv2.CascadeClassifier(rutaDatosEntrenamientoFacial)

class machineLearning():
    def __sonidoAlarma(self, estado):
        if estado == 0:
            #winsound.Beep(): Método que reproduce un sonido tipo Buzzer con cierta frecuencia y duración.
            frecuencia = 2500    #Frecuencia = Hz.
            duracion = 250      #Duración = ms.
            winsound.Beep(frecuencia, duracion)

    def __suenaAlarmaThread(self, estado):
        #threading.Thread(): El constructor de la clase Thread se utiliza para crear un nuevo hilo en el programa,
        #que se ejecutará en paralelo con el hilo principal del programa u otros hilos. Para ello en el
        #constructor se debe declarar una función que se ejecute en el nuevo hilo, sin poner sus paréntesis.
        hiloVision = threading.Thread(target = self.__sonidoAlarma, args = (estado, ))
        #threading.Thread().start(): Una vez creado el nuevo hilo con el constructor de la clase Thread, se
        #inicializará su ejecución con el método start().

```



```

hiloVision.start()

#El reconocimiento facial y la alarma que detecta a un desconocido se detendrá dependiendo del valor del
#parámetro estado,

# - Si estado == 0: Se ejecuta el reconocimiento facial y puede sonar la alarma.
# - Si estado == 1: Se detiene el reconocimiento facial y deja de sonar la alarma.

def reconocimientoFacial(self, estado):

    #cv2.VideoCapture(): Método de python para acceder a la webcam, como parámetro solo se enlista el número
    #de webcams a las que se quiere acceder, empezando a contar desde el índice 0, 1, 2, etc. 0 se le puede
    #pasar entre comillas la ruta de un video para que lo reproduzca.

    camara = cv2.VideoCapture(0)

    while True:

        #.isOpened(): Método que se aplica a un objeto de OpenCV que haya abierto una webcam, este identifica
        #si la ventana de la webcam está abierta o no, para que se realice el análisis de la imagen en el
        #video.

        if not camara.isOpened():

            #Mensaje que se imprimirá en consola cuando no se pueda abrir la webcam del ordenador.

            print("No es posible abrir la cámara")

            exit()

        #Bucle while: Cuando se analice un video, se debe realizar dentro de un bucle while para que se
        #ejecute hasta que la ventana donde se muestra el video se cierre.

        while True:

            #.read(): Método que se aplica a un objeto de OpenCV que haya abierto una webcam, este método
            #obtiene dos resultados:

            # - ret: Variable booleana que puede ser true si hay un video (fotograma) capturado y false si no
            #   hay fotograma que leer, este se usa más que nada para crear un if cuando se quiere leer un
            #   video ya existente en vez de grabar y mostrar uno nuevo o si se quiere asegurar que la webcam
            #   está grabando.

            # - frame: La variable frame devuelve todas las imágenes del video capturado.

            ret, frame = camara.read()

            #Condicional que se ejecuta cuando no se pudo obtener un fotograma.

            if not ret:

                print("No es posible obtener la imagen")

                break

            else:

                #OBTENCIÓN DE COLORES HSV:

                #cvtColor(): Método de la librería OpenCV que recibe como primer parámetro una imagen RGB y
                #en su segundo parámetro recibe un atributo de OpenCV llamado cv2.COLOR_BGR2HSV para
                #convertir una imagen RGB a su formato de colores HSV o cv2.COLOR_BGR2GRAY para obtener su
                #escala de grises.

                imgEscalaGrises = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

                #copy(): Método que crea una copia de la imagen.

                imagenAuxiliar = imgEscalaGrises.copy()

```



```

#objetoCascadeClassifier.detectMultiScale(): Con este método se accede a la base de datos de
#reconocimiento facial de OpenCV para el reconocimiento de rostros, a este se le pasa como
#parámetro la imagen que se quiere analizar, pero antes se debió haber obtenido la escala de
#grises de la imagen y jalado con el método cv.CascadeClassifier() la base de datos de
#reconocimiento de rostros, después para mejorar la detección de rostros se puede indicar un
#factor de escala que se encargue de reducir el tamaño de la imagen en cada búsqueda durante
#la detección de caras y finalmente el mínimo número de vecinos para que una región sea
#considerada una detección válida, donde un valor más alto requiere que más comprobaciones
#correctas para ser tomada en cuenta como una detección correcta. Esto ayuda a eliminar
#detecciones falsas.

caras = clasificacionFacial.detectMultiScale(imgEscalaGrises, 1.3, 5)

#Bucle for que dibuja un rectángulo alrededor del rostro identificado, para ello debe
#reconocer el ancho y alto del rostro.
color_rectangulo = (176, 224, 17) #El vector de color BGR se guarda en una lista.
grosor_rectangulo = 2 #Grosor del rectángulo en píxeles
for (x, y, ancho, altura) in caras:
    #Recordemos que la imagen en realidad es una matriz, entonces para acceder a solo una
    #parte de ella, se indica que de la matriz, osea del numpy array, solo se tome un rango
    #de valores que vayan desde x hasta ancho en la parte horizontal de las 3 capas y de las
    #coordenadas y hasta altura en la parte vertical de las 3 capas que conforman la imagen
    #digital.
    cara = imagenAuxiliar[y:y + altura, x:x + ancho]
    #resize(): Lo que hace este método es cambiar el tamaño de una imagen, para ello recibe
    #como primer parámetro la variable que almacena la imagen y como segundo parámetro indica
    #la magnitud de la nueva imagen, como segundo parámetro se le da un nuevo tamaño de
    #150x150 píxeles y finalmente el parámetro opcional interpolation sirve para ajustar los
    #valores de píxeles en la matriz de la nueva imagen para que no se pierda su calidad y
    #esencia.
    cara = cv2.resize(cara, (150, 150), interpolation = cv2.INTER_CUBIC)
    #cv2.face.LBPHFaceRecognizer_create().read().predict(): El método predict() que se aplica
    #al objeto de reconocimiento facial LBPH (Local Binary Patterns Histograms) y previamente
    #ha sido alimentado con el modelo de reconocimiento facial de un archivo xml propio,
    #sirve para analizar el fotograma (imagen del video) captado y de esta forma identificar
    #si el rostro es el del usuario a través de un rango que puede adoptar valores de 0 a 80
    #cuando si se reconozca el rostro del usuario, si no se reconoce el valor será mayor a 80.
    resultadoFacial = reconocimientoFacial.predict(cara)
    print("Resultado de reconocimiento facial:\n" + str(resultadoFacial))

#cv2.putText(): Método utilizado para agregar texto a una imagen o un fotograma de video,
#para ello recibe como parámetros la imagen donde se quiere colocar, el texto, las
#coordenadas donde se posicionará, el tipo de letra, la escala de su tamaño, su color,
#grosor, y tipo de línea para dibujar las letras, que usualmente es el cv2.LINE_AA.
tipoLetra = cv2.FONT_HERSHEY_PLAIN #Tipo de letra que indica el nombre del usuario.
cv2.putText(frame, f"{resultadoFacial}", (x, y-5), tipoLetra, 1.1, (0, 0, 200), 1, cv2.LINE_AA)

```



```

#Si el resultado del reconocimiento facial se encuentra entre 0 y 80 el rostro fue
#identificado, pero si su resultado es mayor a 80, significa que el rostro no fue
#reconocido.
if (resultadoFacial[1] < 85):
    #Para extraer el nombre del usuario se recorrerá el directorio que contiene las
    #imágenes de reconocimiento facial, ya que al crear esas carpetas, a través de la
    #variable usuario se indicó su nombre y este se reconoce con el valor del
    #identificador del resultadoFacial porque está asociado de igual manera a los id de
    #sus datos.
    cv2.putText(frame, f"{dataImagenesUsuarios[resultadoFacial[0]]}", (x, y-25), tipoLetra, 2, (200,
200, 200), 1, cv2.LINE_AA)

    #cv.rectangle(): Con este método perteneciente a la librería OpenCV se puede crear un
    #rectángulo, en él se indica primero en que imagen se va a dibujar la figura, luego
    #las coordenadas iniciales x,y de la imagen en donde se va a dibujar la esquina
    #superior izquierda del rectángulo, seguido de las coordenadas x2,y2 que consideran
    #el tamaño de la imagen, su color y si se quiere que sea hueco o sólido el rectángulo,
    #si se buscara que fuera sólido se pondría un -1, si fuera hueco, en esta parte se
    #indicaría el grosor del borde del rectángulo en pixeles.
    cv2.rectangle(frame, (x, y), (x + ancho, y + altura), color_rectangulo, grosor_rectangulo)
else:
    #__suenaAlarmaThread(): Función propia que recibe como parámetro el estado del
    #reconocimiento facial y dependiendo del valor de este se activa o desactiva la
    #alarma.
    self.__suenaAlarmaThread(estado)

    #Si no se reconoció al usuario, se hará sonar una alarma y se pondrá el texto de
    #usuario desconocido en la imagen del video, además de que se pintará de igual manera
    #un rectángulo que rodee su rostro.
    cv2.putText(frame, "Usuario desconocido", (x, y-20), tipoLetra, 1.1, (0, 0, 255), 1, cv2.LINE_AA)
    cv2.rectangle(frame, (x, y), (x + ancho, y + altura), (0, 0, 255), 4)

#imshow(): Con el método se puede mostrar una imagen, el primer parámetro es el nombre de la
#ventana donde aparecerá la imagen y el segundo parámetro es la imagen recopilada.
cv2.imshow('Reconocimiento facial', frame)

#waitKey(): Método que permite a los usuarios mostrar una ventana durante un número de
#milisegundos determinados si su parámetro es un número mayor a 1 o hasta que se presione
#cualquier tecla. Toma un tiempo en milisegundos como parámetro y espera el tiempo dado para
#destruir la ventana:
# - Si se pasa 0 como argumento, espera hasta que se presiona cualquier tecla.
# - Si se pasa 1 como argumento, espera hasta que se presione una tecla específica para
#   cerrar la ventana.

#El método waitKey() está monitoreando si se presiona una tecla o no, este modo se activa si
#se le pasa como parámetro un número 1, de esta manera:
# - Si no se presiona una tecla retorna un valor -1 (False).
# - Si se presiona una tecla, retorna un valor ASCII correspondiente a la tecla que se

```



```

# oprimió, con el método ord() podemos indicar el código ASCII de la letra del teclado que
# indiquemos en su parámetro o simplemente en un condicional un valor diferente a -1 o 0
# se considera como True.
#Se cierra el programa cuando se presione cualquier tecla y el valor de estado sea 1.
if (cv2.waitKey(1) == ord('r')):
    estado = 1
    #cv2.VideoCapture().release(): Con este método se libera la webcam, tanto en software
    #como en hardware.
    camara.release()
    #destroyAllWindows(): Función que permite a los usuarios destruir todas las ventanas en
    #cualquier momento. No toma ningún parámetro y no devuelve nada, esto se incluye para que
    #al cerrar la ventana después del método waitKey() se destruyan para poder utilizarse en
    #otra cosa.
    cv2.destroyAllWindows()
    #subprocess.call(): Método que sirve para ejecutar un comando en la consola del sistema
    #operativo Windows, para ello recibe los siguientes parámetros:
    # - args: Indica el comando que se va a ejecutar en el sistema operativo Windows.
    # - taskill: Este comando se utiliza para matar la ejecución de un thread o proceso
    # perteneciente a cierto programa o ejecutable del ordenador.
    # - /IM python.exe: Con esta instrucción se indica el programa que debe detener la
    # instrucción taskill.
    # - /F: Con este argumento se le dice a la consola que el proceso debe ser finalizado
    # a la fuerza, sin preguntar al usuario para confirmación.
    # - shell: Si este parámetro se indica como True, el comando se ejecutará a través de una
    # consola shell del sistema operativo.
    subprocess.call(args = f"taskill /IM python.exe /F", shell = True)
    break

```

Función: 1_DatosFaciales

```

#IMPORTACIÓN DE LIBRERÍAS:
import cv2      #Librería OpenCV: Sirve para todo tipo de operaciones de visión artificial
import os       #os: Librería que permite acceder a funciones y métodos relacionados con el sistema operativo.
import imutils  #imutils: Librería que se utiliza para simplificar y acelerar el procesamiento de imágenes.

#Este programa lo que hará es recibir el video de alguna persona que se quiera reconocer, luego de este video
#extraerá varias fotos pequeñas de su rostro en distintos ángulos y así después se entrenará al programa para
#reconocer la cara de cada persona, ya que se haya llevado a cabo el reconocimiento facial se creará una
#carpeta con su nombre, si se busca reconocer una persona nueva, solo se debe cargar otro video y cambiar el
#nombre de la variable usuario para que cree una nueva carpeta de datos faciales.

#Creación de la ruta donde guardará el programa los datos faciales del usuario para que lo reconozca.
usuario = "di_cer0"
ruta = "C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/UsuariosReconocimientoFacial"

```



```

usuarioPath = ruta + "/" + usuario

#os.path.exists(): Método que comprueba si un directorio existe en mi ordenador.
#os.makedirs(): Método que crea un nuevo directorio en mi ordenador.
if not os.path.exists(usuarioPath): #Evalúa si existe el path con la cara del usuario y sino lo crea.
    os.makedirs(usuarioPath)

#cv2.VideoCapture(): Método de python para acceder a la webcam, como parámetro solo se enlista el número de
#webcams a las que se quiere acceder, empezando a contar desde el índice 0, 1, 2, etc. 0 se le puede pasar
#entre comillas la ruta de un video para que lo reproduzca.
#Este será el video que el reconocimiento facial utilizará para reconocer un rostro, pero para que funcione
#mejor se recomienda tomar el video con una alta calidad, buena iluminación y realizar la grabación en el
#mismo lugar donde se vaya a utilizar el asistente virtual.
rostro = cv2.VideoCapture("C:/Users/diego/Pictures/Camera Roll/Rostro_di_cer0.mp4")

#RECONOCIMIENTO DE ROSTROS CON OPENCV:
#Documentación de la base de datos de OpenCV para realizar el reconocimiento facial:
#https://github.com/opencv/opencv/tree/master
#HAAR CASCADES: Algoritmo de reconocimiento facial, para poder hacer el reconocimiento facial se debe hacer la
#extracción de la base de datos de haarcascade, de sus archivos xml es de donde se saca la información para
#realizar un correcto reconocimiento facial separando el rostro de una persona de su fondo o demás elementos
#del video o imagen, en específico se utiliza el archivo haarcascade_frontalface_default.xml que se encuentra
#en el siguiente link, es recomendable tener una copia local del archivo para el reconocimiento facial por
#cualquier tema de conexión, actualización de la base de datos, etc. Para identificar cada parte distinta del
#rostro se deben usar diferentes bases de datos dentro de la documentación de OpenCV.

#cv.CascadeClassifier(): Este es un clasificador en cascada, osea que dependiendo de ciertas características
#del reconocimiento facial, puede ir segmentando el video o imagen hasta identificar un rostro, ya sea de
#hombre, mujer, perro, gato, etc. porque cada uno tiene características especiales, para que funcione esto,
#se debe indicar como parámetro del método la base de datos descargada de la documentación de Github de OpenCV
#con extensión xml.
#https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
#Para la identificación de rostros, específicamente se utiliza la base de datos llamada
#haarcascade_frontalface_default.xml:
#https://github.com/opencv/opencv/tree/4.x/data/haarcascades
rutaDatosEntrenamientoFacial = "C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/API_Keys/haarcascade_frontalface_default.xml"
clasificacionFacial = cv2.CascadeClassifier(rutaDatosEntrenamientoFacial)
numeroCara = 0

#.isOpened(): Método que se aplica a un objeto de OpenCV que haya abierto una webcam, este identifica si la
#ventana de la webcam está abierta o no, para que se realice el análisis de la imagen en el video.
if not rostro.isOpened():
    #Mensaje que se imprimirá en consola cuando no se pueda abrir la webcam del ordenador.
    print("No es posible abrir la cámara")

```



```

    exit()

#Bucle while: Cuando se analice un video, se debe realizar dentro de un bucle while para que se ejecute
#hasta que la ventana donde se muestra el video se cierre.
while True:

    #.read(): Método que se aplica a un objeto de OpenCV que haya abierto una webcam, este método obtiene
    #dos resultados:

    # - ret: Variable booleana que puede ser true si hay un video (fotograma) capturado y false si no hay
    #   fotograma que leer, este se usa más que nada para crear un if cuando se quiere leer un video ya
    #   existente en vez de grabar y mostrar uno nuevo o si se quiere asegurar que la webcam está grabando.
    # - frame: La variable frame devuelve el video capturado.

    ret, frame = rostro.read()

    #Condicional que se ejecuta cuando no se pudo obtener un fotograma.
    if not ret:

        print("Datos Faciales Cargados...")

        break

    #imutils.resize(): Método para redimensionar una imagen en Python.
    frame = imutils.resize(frame, width = 640)

    #cvtColor(): Método de la librería OpenCV que recibe como primer parámetro una imagen RGB y en su
    #segundo parámetro recibe un atributo de OpenCV llamado cv2.COLOR_BGR2HSV para convertir una imagen
    #RGB a su formato de colores HSV, escala de grises, etc.
    img_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    #copy(): Método que crea una copia de la imagen.
    imagenAuxiliar = frame.copy()

    #objetoCascadeClassifier.detectMultiScale(): Con este método se accede a la base de datos de
    #reconocimiento facial de OpenCV para el reconocimiento de rostros, a este se le pasa como parámetro
    #la imagen que se quiere analizar, pero antes se debió haber obtenido la escala de grises de la imagen
    #y jalado con el método cv.CascadeClassifier() la base de datos de reconocimiento de rostros, después
    #para mejorar la detección de rostros se puede indicar un factor de escala que se encargue de reducir el
    #tamaño de la imagen en cada búsqueda durante la detección de caras y finalmente el mínimo número de
    #vecinos para que una región sea considerada una detección válida, donde un valor más alto requiere que
    #más comprobaciones correctas para ser considerada como una detección válida. Esto ayuda a eliminar
    #detecciones falsas.

    caras = clasificacionFacial.detectMultiScale(img_gray, 1.3, 7)

    #Bucle for que dibuja un rectángulo alrededor del rostro identificado, para ello debe reconocer el ancho
    #y alto del rostro.
    color_rectangulo = (176, 224, 17) #El color se guarda en una lista, se maneja como BGR en OpenCV
    grosor_rectangulo = 2 #Grosor del rectángulo en píxeles
    for (x, y, ancho, altura) in caras:

        #cv.rectangle(): Con este método perteneciente a la librería OpenCV se puede crear un rectángulo, en
        #él se indica primero en que imagen obtenida con el método imread() se va a dibujar la figura, luego
        #las coordenadas iniciales x,y de la imagen en donde se va a dibujar la esquina superior izquierda del
        #rectángulo, seguido de las coordenadas x2,y2 que consideran el tamaño de la imagen, su color y si se
        #quiere que sea hueco o sólido el rectángulo, si se buscara que fuera sólido se pondría un -1, si

```




```

#fuera hueco, en esta parte se indicaría el grosor del borde del rectángulo en píxeles.
cv2.rectangle(frame, (x, y), (x + ancho, y + altura), color_rectangulo, grosor_rectangulo)

#Recordemos que la imagen en realidad es una matriz, entonces para acceder a solo una parte de ella,
#se indica que de la matriz, osea del numpy array, solo se tome un rango de valores que vayan desde x
#hasta ancho en la parte horizontal de las 3 capas y de las coordenadas y hasta altura en la parte
#vertical de las 3 capas que conforman la imagen digital.

cara = imagenAuxiliar[y:y + altura, x:x + ancho]

#resize(): Lo que hace este método es cambiar el tamaño de una imagen, para ello recibe como primer
#parámetro la variable que almacena la imagen y como segundo parámetro indica la magnitud de la nueva
#imagen, como segundo parámetro se le da un nuevo tamaño de 150x150 píxeles y finalmente el parámetro
#opcional interpolation sirve para ajustar los valores de píxeles en la matriz de la nueva imagen
#para que no se pierda su calidad y esencia.

cara = cv2.resize(cara, (150, 150), interpolation = cv2.INTER_CUBIC)

#imwrite(): Método de la librería OpenCV que sirve para guardar una imagen, como primer parámetro
#tengo que poner el nombre con el que se guardará la imagen con todo y extensión y en el segundo debo
#poner la variable de la que se extrae la imagen.

cv2.imwrite(usuarioPath + f"/cara_{numeroCara}.jpg", cara)

numeroCara += 1

#imshow(): Con el método se puede mostrar una imagen, el primer parámetro es el nombre de la ventana donde
#aparecerá la imagen y el segundo parámetro es la imagen recopilada.

cv2.imshow('Datos de reconocimiento facial', frame)

#waitKey(): Método que permite a los usuarios mostrar una ventana durante un número de milisegundos
#determinados si su parámetro es un número mayor a 1 o hasta que se presione cualquier tecla. Toma
#tiempo en milisegundos como parámetro y espera el tiempo dado para destruir la ventana:

# - Si se pasa 0 como argumento, espera hasta que se presione cualquier tecla.
# - Si se pasa 1 como argumento, espera hasta que se presione una tecla específica para cerrar la
#   ventana.

#El método waitKey() está monitoreando si se presiona una tecla o no, este modo se activa si se le
#pasa como parámetro un número 1, de esta manera:

# - Si no se presiona una tecla retorna un valor -1
# - Si se presiona una tecla, retorna un valor ASCII correspondiente a la tecla que se oprimió, con
#   el método ord() podemos indicar el código ASCII de la letra del teclado que indiquemos en su
#   parámetro.

if (cv2.waitKey(1) == ord('r')) or (numeroCara >= 300): #Se cierra el programa cuando se presione la letra q
    #cv2.VideoCapture().release(): Con este método se libera la webcam, tanto en software como en
    #hardware.

    rostro.release()

#destroyAllWindows(): Función que permite a los usuarios destruir todas las ventanas en
#cualquier momento. No toma ningún parámetro y no devuelve nada, esto se incluye para que al
#cerrar la ventana después del método waitKey() se destruyan para poder utilizarse en otra cosa.

cv2.destroyAllWindows()

break

```



Función: 2_EntrenamientoFacial

```
#IMPORTACIÓN DE LIBRERÍAS:
import cv2      #Librería OpenCV: Sirve para todo tipo de operaciones de visión artificial
import os       #os: Librería que permite acceder a funciones y métodos relacionados con el sistema operativo.
import numpy as np #numpy: Librería que realiza operaciones matemáticas complejas.

#Este programa lo que hará es recibir el video de alguna persona que se quiera reconocer, luego de este video
#extraerá varias fotos pequeñas de su rostro en distintos ángulos y así después se entrenará al programa para
#reconocer la cara de cada persona, ya que se haya llevado a cabo el reconocimiento facial se creará una
#carpeta con su nombre, si se busca reconocer una persona nueva, solo se debe cargar otro video y cambiar el
#nombre de la variable usuario para que cree una nueva carpeta de datos faciales.

#Creación de la ruta donde guardará el programa los datos faciales del usuario para que lo reconozca.
rutaUsuarios = "C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/UsuariosReconocimientoFacial"

#os.listdir(): Método que extrae todos los nombres de los archivos y carpetas contenidos en un directorio.
dataUsuarios = os.listdir(rutaUsuarios)
print(dataUsuarios)
carasUsuarios = []
etiquetasCaras = []
etiquetaNumerica = 0

#Bucle for que recorre todas las imágenes de todos los usuarios que se encuentren en la carpeta de
#ReconocimientoFacial_AsistenteVirtual_MarkI, para ello se debió haber ejecutado mínimo una vez el archivo
#1_DatosFaciales.py.
for persona in dataUsuarios:
    rutaPersona = rutaUsuarios + "/" + persona
    print("Leyendo la carpeta de un usuario")
    for imagenCara in os.listdir(rutaPersona):
        print("Caras: " + persona + "/" + imagenCara)
        #Asignación de un id o etiqueta numérica que después se relacionará a cada imagen de la cara del
        #usuario.
        etiquetasCaras.append(etiquetaNumerica)

        #imread(): Sirve para leer la imagen de un directorio especificado, si en el segundo parámetro se pone
        #un 1, la imagen se obtendrá en forma de matriz 3D RGB y si se pone un 0 se obtendrá en escala de
        #grises, obteniendo una matriz 2D.
        imgCara = cv2.imread(rutaPersona + "/" + imagenCara, 0)
        carasUsuarios.append(imgCara)
        etiquetaNumerica += 1

#cv2.face.LBPHFaceRecognizer_create(): Este método crea un objeto de reconocimiento de rostros utilizando el
#algoritmo de reconocimiento de patrones basado en histogramas locales de patrones binarios (Local Binary
#Patterns Histograms, LBPH) de OpenCV que se utiliza para reconocer caras en imágenes y luego crear un
#histograma de patrones binarios, el cual será utilizado para identificar cada rostro una vez que haya sido
#entrenado con un conjunto de imágenes etiquetadas de una cara en específico.
```



```

reconocimientoUsuario = cv2.face.LBPHFaceRecognizer_create()
print("Ingresando nuevo usuario al reconocimiento facial... bip bup...")
#cv2.face.LBPHFaceRecognizer_create().train(): Este método se utiliza para entrenar el objeto LBPH con un
#conjunto de datos de entrenamiento que consiste en imágenes de rostros etiquetadas con un id correspondiente
#a cada usuario. En su primer parámetro recibe una lista con los datos de las imágenes del entrenamiento y en
#el segundo un numpy array que contenga una lista de las etiquetas correspondientes a cada imagen.
reconocimientoUsuario.train(carasUsuarios, np.array(etiquetasCaras))

#cv2.face.LBPHFaceRecognizer_create().write(): Método que se utiliza para guardar un modelo de reconocimiento
#de rostros previamente entrenado en un archivo con extensión XML, cuyo nombre y extensión se debe indicar
#como su parámetro.
rutaModeloEntrenado = "C:/Users/diego/OneDrive/Documents/The_MechaBible/p_Python_ESP/5.-Inteligencia
Artificial/ReconocimientoFacial_AsistenteVirtual_MarkI/modeloReconocimientoFacial.xml"
reconocimientoUsuario.write(rutaModeloEntrenado)
print("Modelo de reconocimiento facial entrenado y almacenado en un archivo xml...")

```

Referencias

AMP Tech, “Intro a LangChain: Construye sobre LLMs/GPT4”, 2023 [Online], Available: https://www.youtube.com/watch?v=GoSbWLO_eGI

AMP Tech, “Langchain 1: Modelos y Prompts”, 2023 [Online], Available: <https://www.youtube.com/watch?v=qx3adFfbJRs>

AMP Tech, “LangChain: GPT4 ahora no olvidará nada”, 2023 [Online], Available: <https://www.youtube.com/watch?v=7xpykL0jAEA>

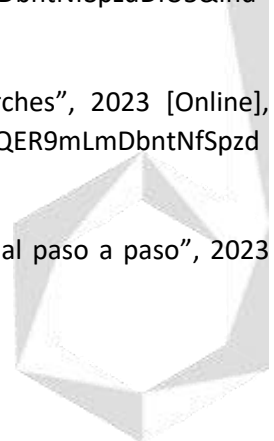
AMP Tech, “LangChain 3: Cadenas con GPT4”, 2023 [Online], Available: <https://www.youtube.com/watch?v=m1O6PJjEWnY>

Rabbitmetrics, “LangChain Explained in 13 Minutes | QuickStart Tutorial for Beginners”, 2023 [Online], Available: <https://www.youtube.com/watch?v=aywZrzNaKjs>

Greg Kamradt (Data Indy), “LangChain 101: Ask Questions On Your Custom (or Private) Files + Chat GPT”, 2023 [Online], Available: <https://www.youtube.com/watch?v=EnT-ZTrcPrg&list=PLqZXAKvF1bPNQER9mLmDbntNfSpzdDIU5&index=11>

Greg Kamradt (Data Indy), “LangChain 101: Agents Overview + Google Searches”, 2023 [Online], Available: <https://www.youtube.com/watch?v=Jq9Sf68ozk0&list=PLqZXAKvF1bPNQER9mLmDbntNfSpzdDIU5&index=5>

Platzi, “Cómo transcribir audio y video gratis usando Whisper y Python | Tutorial paso a paso”, 2023 [Online], Available: <https://www.youtube.com/watch?v=SL6qw9-9NEQ>



Daniiee, “Asistente virtual con Python”, 2023 [Online], Available:
<https://www.youtube.com/playlist?list=PLQuweJwUyMhXzZ1YxibQ3OyaYMWz8Hqam>

Daniiee, “Tutoriales de Python”, 2021 [Online], Available:
<https://www.youtube.com/playlist?list=PLQuweJwUyMhXv3F5RTOMO1dAF2UwC1zde>

