# Reporte de avances

Dennis Masso Macías

Yuluka Gigante Muriel

# Contexto del problema

Actualmente existe una gran cantidad de asistentes virtuales que tienen, en su mayoría, las mismas funcionalidades entre sí. Sin embargo, aparte de las funcionalidades, tienen el factor común de no ser capaces de ofrecer conversaciones cuya naturalidad se asemeje a la que se podría tener con otra persona.

Además de estos asistentes convencionales, también existen otros como ChatGPT, BingChat, Bard, entre otros. Sin embargo, estos carecen de la comodidad que sí ofrecen los otros, pues su acceso tiene una mayor cantidad de pasos intermedios y, aunque ofrecen funcionalidades más avanzadas, estas no se encuentran integradas en el mismo lugar, por lo que su uso se hace más engorroso.

Es con base en esto, e inspirados por la Inteligencia Artificial de fantasía de los comics, Jarvis, que decidimos crear nuestro propio asistente virtual que combine lo mejor de ambos mundos. Por un lado, se busca la posibilidad de tener las conversaciones racionales y naturales que nos ofrecen las IA como ChatGPT, a la vez que la comodidad de la comunicación por voz y acceso rápido que podríamos tener con Syri o Google Assistant.

# Proceso de investigación

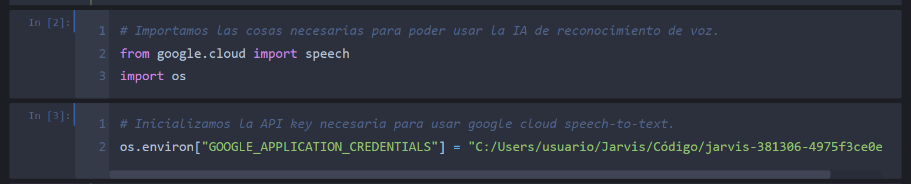
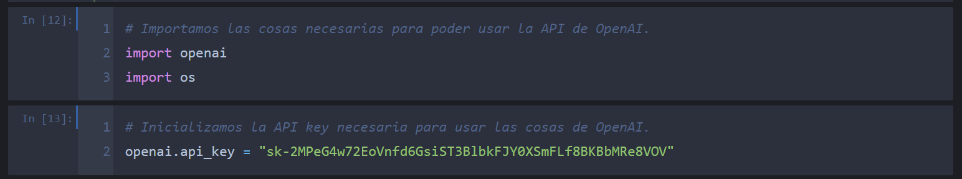
Para el desarrollo de nuestro proyecto, y puesto que perseguíamos un objetivo muy ambicioso, decidimos que la mejor opción era la de utilizar Interfaces de Programación de Aplicaciones (APIs).

Sabiendo esto, lo siguiente fue la búsqueda de APIs de reconocimiento de voz (Speech to text), conversión de texto a voz (Text to speech), y chat. Finalmente, las APIs seleccionadas fueron: Google Cloud Speech-To-Text, Google Cloud Text-To-Speech, y OpenAI API.

El paso siguiente fue documentarnos en la cuestión del funcionamiento de estas. Para esto, nuestras principales fuentes fueron las documentaciones ofrecidas por Google y OpenAI, además de foros de preguntas y respuestas como Stack Overflow. Los enlaces a las documentaciones son los siguientes:

* [Google Cloud Speech-To-Text](https://cloud.google.com/speech-to-text/docs/before-you-begin?hl=es-419): De esta, se desprenden un gran conjunto de otras guías en las que se explican distintas partes del uso de la API.
* [Google Cloud Text-To-Speech](https://cloud.google.com/text-to-speech/docs/before-you-begin?hl=es-419): Igual que con la anterior, esta guía contiene varias más guías que instruyen en el uso de otras partes de la API.
* [OpenAI API](https://platform.openai.com/docs/api-reference/introduction?lang=python): En esta documentación se incluyen las instrucciones del uso de la API para todas las funciones que ofrece OpenAI, tales como Chat, Completions, Image Creation, entre otras.

# Diseño del modelo

****Para hacer el desarrollo de nuestro proyecto, usamos Python y notebooks de Jupyter Notebook y configuramos las APIs para hacer solicitudes a las mismas.

****Además de esto, creamos distintas funciones con el objetivo de interactuar con etas APIs.

# Desarrollo y validación

El proceso de desarrollo estuvo un poco interrumpido por algunas complicaciones con respecto al uso de las APIs y de las librerías de Python desconocidas hasta ese momento. Sin embargo, una vez comprendido el funcionamiento de todo, las cosas fluyeron de forma amena.

La última complicación que tuvimos fue con respecto a que OpenAI restiró la capa gratuita para el uso de su API, por lo que tuvimos que pagar para seguir con el proyecto.

Con respecto a la validación, todos los modelos usados responden de manera satisfactoria siempre. La transformación de voz a texto únicamente omite los signos de puntuación, pero el resto lo detecta a la perfección.

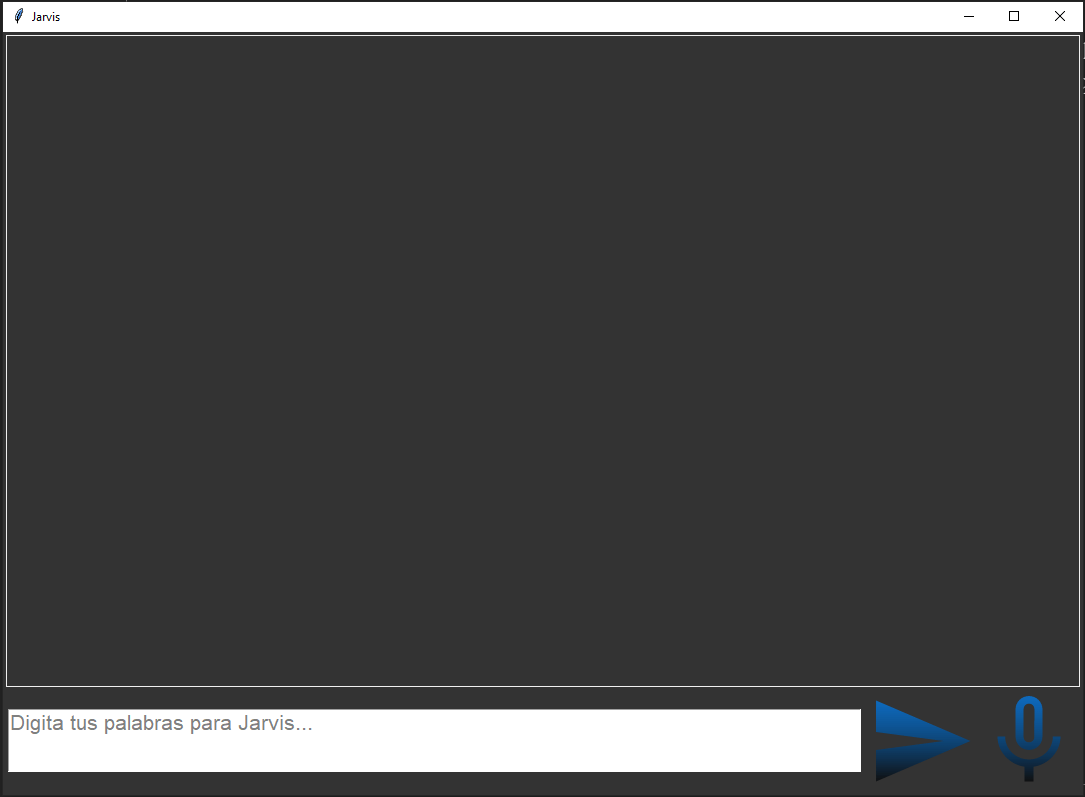
Los tiempos de espera de respuesta se incrementan un poco al usar el ejecutable en lugar de la compilación directa, sin embargo, no son nada del otro mundo y sigue siendo una aplicación por completo funcional.

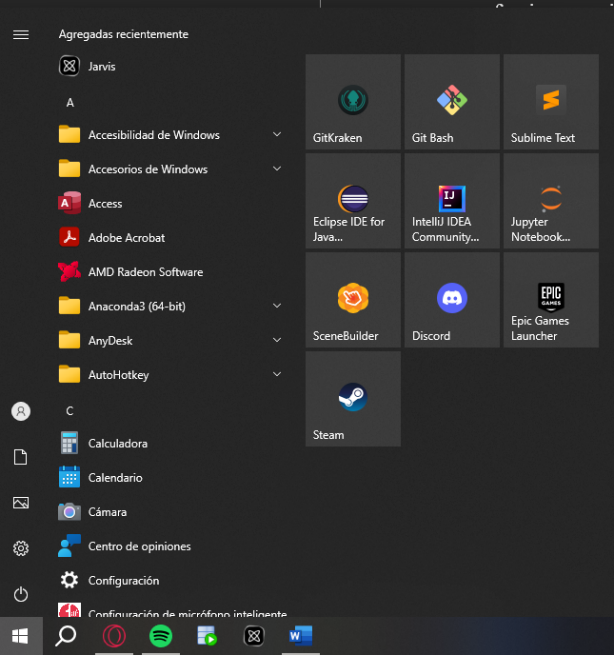
# Despliegue

Para el despliegue decidimos hacer, desde el principio, una aplicación de escritorio que ofreciese una interfaz gráfica óptima para la interacción del usuario con la aplicación.

La razón de hacer el despliegue mediante una aplicación de escritorio es la de eliminar todos los pasos intermedios necesarios para el acceso a los asistentes como ChatGPT.

Por nuestra parte, no solo dejamos la aplicación para ser ejecutada desde el compilador de Jupyter Notebook, sino que convertimos la aplicación en un ejecutable funcional que puede ser accedido como cualquier otra aplicación del sistema.





ww