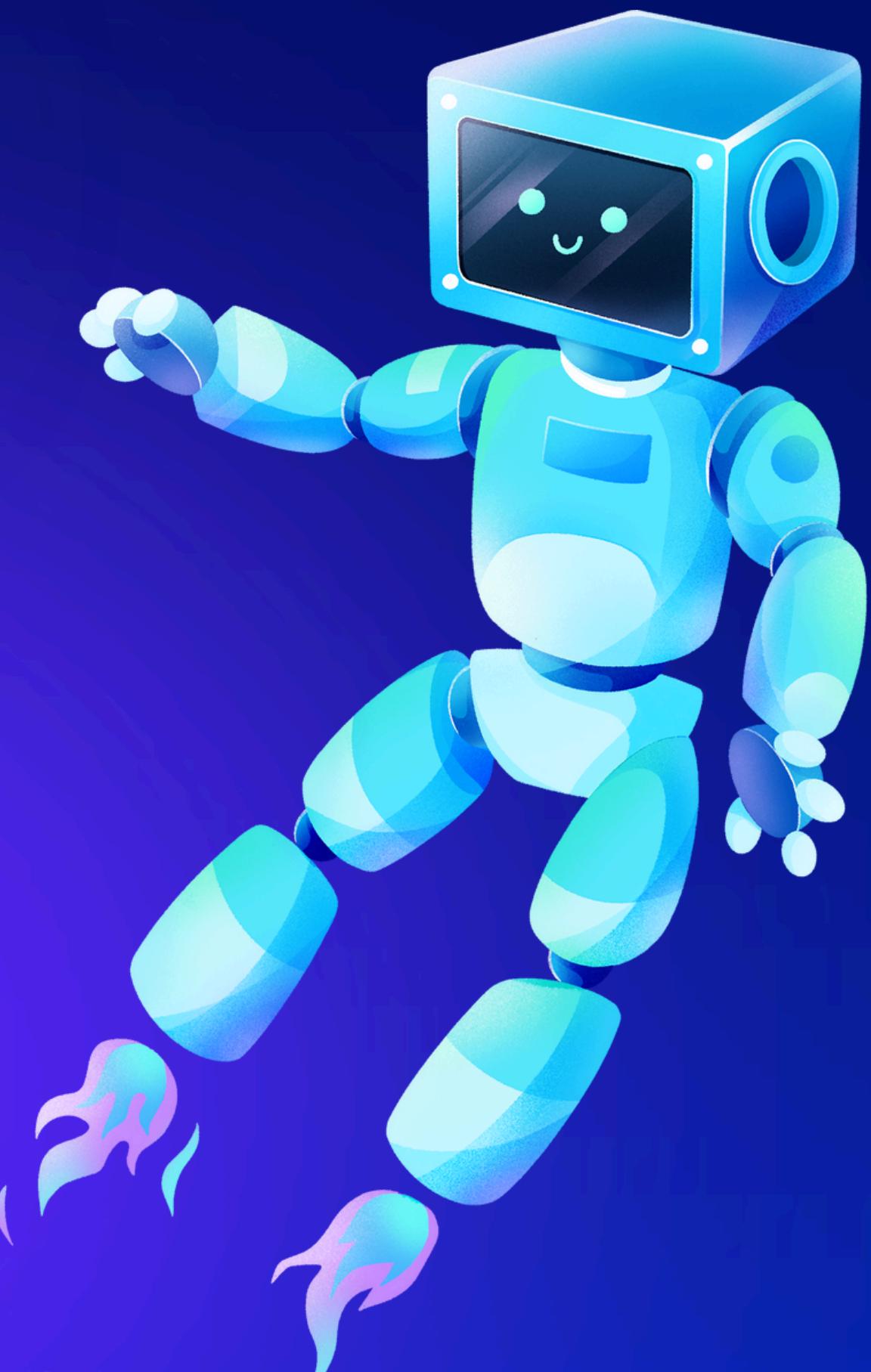


# CHATCARE

Estado del proyecto

29/04/2024

Integrantes:  
Gabriele Petroni  
Ricardo Palomares  
Jesús Espadas





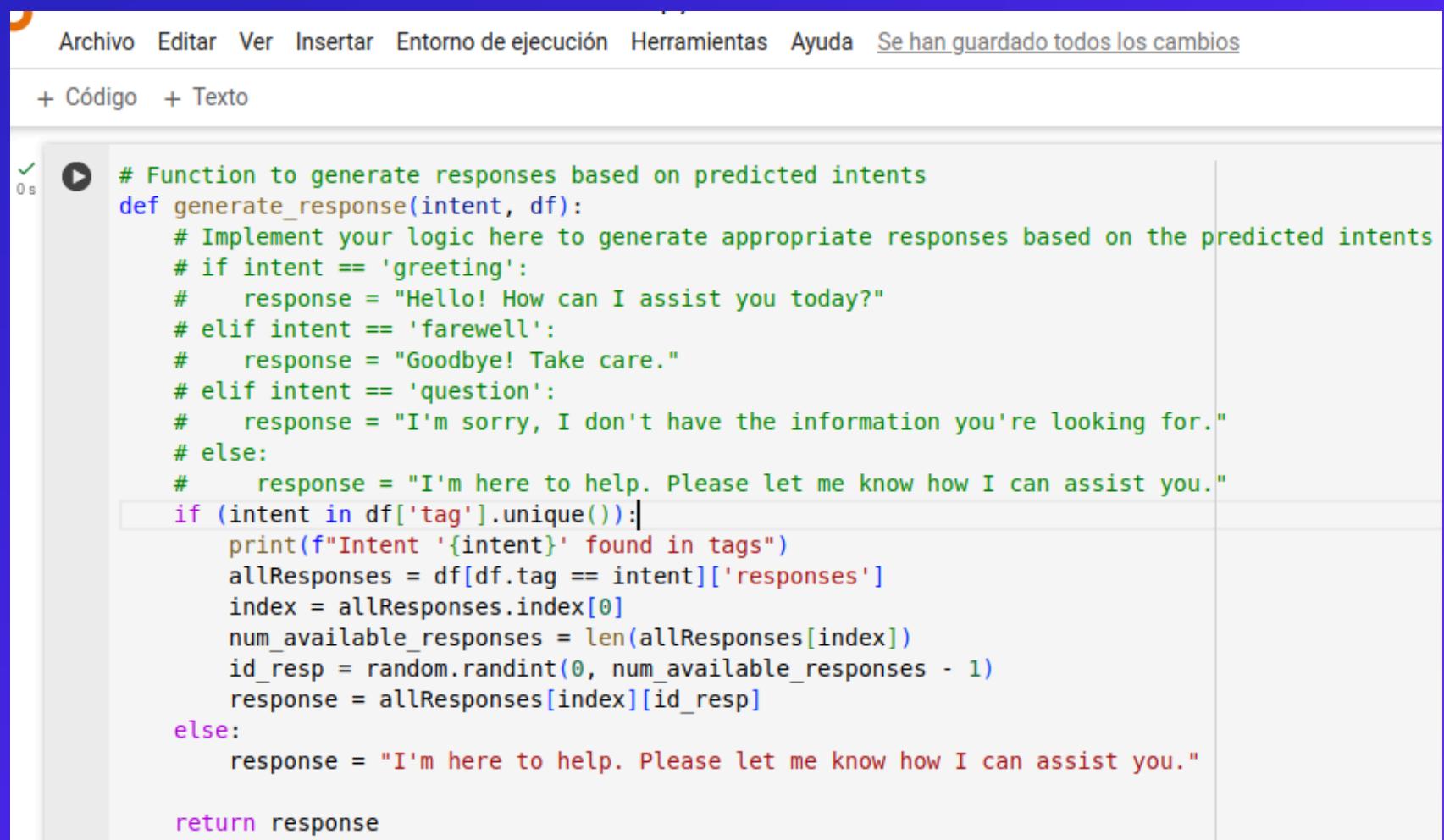
# CONTENIDO

- NoteBook ChatBot de partida.
- Via de desarrollo 1: uso de múltiples datasets alternativos.
- Via de desarrollo 2: etiquetado de muestras adicionales.
- Fine-Tuning.
- Resumen
- Ejemplo.



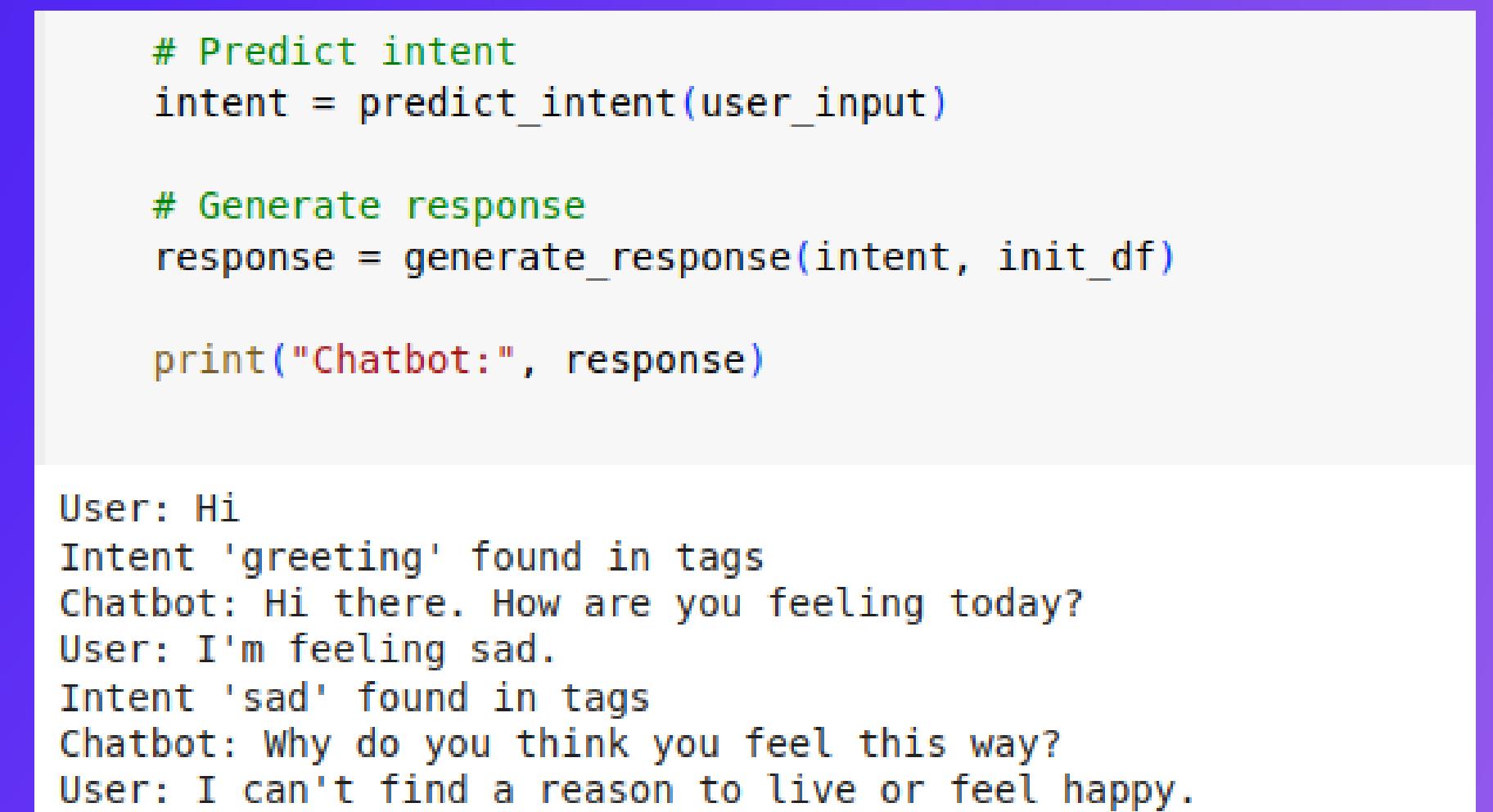
# NOTEBOOK CHATBOT DE PARTIDA.

Enlace al chatbot original



The screenshot shows a Jupyter Notebook interface with a toolbar at the top. The code cell contains a function named `generate_response` which takes a DataFrame `df` as input. The function logic includes handling for 'greeting', 'farewell', 'question', and general cases where it prints a message and randomly selects a response from the DataFrame. It also handles the case where no intent is found in the tags.

```
# Function to generate responses based on predicted intents
def generate_response(intent, df):
    # Implement your logic here to generate appropriate responses based on the predicted intents
    # if intent == 'greeting':
    #     response = "Hello! How can I assist you today?"
    # elif intent == 'farewell':
    #     response = "Goodbye! Take care."
    # elif intent == 'question':
    #     response = "I'm sorry, I don't have the information you're looking for."
    # else:
    #     response = "I'm here to help. Please let me know how I can assist you."
    if (intent in df['tag'].unique()):
        print(f"Intent '{intent}' found in tags")
        allResponses = df[df.tag == intent]['responses']
        index = allResponses.index[0]
        num_available_responses = len(allResponses[index])
        id_resp = random.randint(0, num_available_responses - 1)
        response = allResponses[index][id_resp]
    else:
        response = "I'm here to help. Please let me know how I can assist you."
    return response
```



The screenshot shows a Jupyter Notebook interface with a toolbar at the top. The code cell contains logic to predict an intent using `predict_intent` and then generate a response using `generate_response`. The generated response is then printed.

```
# Predict intent
intent = predict_intent(user_input)

# Generate response
response = generate_response(intent, init_df)

print("Chatbot:", response)
```

The transcript below shows a user interaction with the chatbot:

User: Hi  
Intent 'greeting' found in tags  
Chatbot: Hi there. How are you feeling today?  
User: I'm feeling sad.  
Intent 'sad' found in tags  
Chatbot: Why do you think you feel this way?  
User: I can't find a reason to live or feel happy.

# VIA DE DESARROLLO 1: USO DE MÚLTIPLES DATASETS ALTERNATIVOS .

- Crear varios ChatBots con diferentes DataSets.
- Modificar y combinar los DataSets de diferentes formas buscando el ChatBot que mejor funcione.
- Obtener al menos 100 conversaciones para pasarlas por Fine-Tuning



# VÍA DE DESARROLLO 2: ETIQUETADO DE MUESTRAS ADICIONALES



## Obtención de datasets adicionales

Se han buscado y descargado datasets relacionados con salud mental adicionales desde, fundamentalmente, HuggingFace. Se han buscado datasets que representaran diálogos entre pacientes y psicólogos, descartando glosarios y similares.

## Algoritmo

Partiendo del chatbot original, se siguen estos pasos:

- Se carga el dataset original y otro adicional.
- Se crea el modelo con el dataset original.
- Se inyectan inputs del dataset extra como prompts del usuario.
- Se muestra la clasificación hecha por el modelo y se permite aceptarla, corregirla o añadir nuevas etiquetas con los inputs.

Por tiempo, no será posible agregar todos los datasets (diferencia con vía de desarrollo 1). Finalmente, se volcarán conversaciones.

# FINE-TUNING

La etapa final del proyecto consiste en la realización de un chatbot a partir de un modelo GPT (OpenAI) con aplicación de fine-tuning\*.

Utilizamos los modelos entrenados en python para realizar el dataset necesario (patrón respuestas) que una vez revisado, será parte integrante del dataset de fine-tuning.

*\*Para acceder a los servicios de OpenAI es necesario obtener una autorización por parte de Microsoft. En caso de no obtener dicha autorización, optaremos por una alternativa.*

# RESUMEN

1. Bots en python que nos permiten realizar patrones de respuesta (pareja consulta/respuesta)



2. Revisamos los datos generados para corregirlos si fuera necesarios y para convertirlos al formato correcto (JSON)

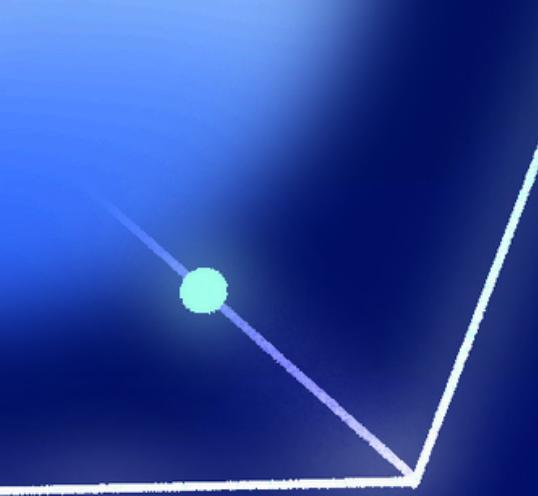


4. El modelo GPT entrenado se queda almacenando en la nube en un entorno Azure y es accesible a través de sus propias APIs



3. Realizamos un programa python para comunicar con las APIs de OpenAI y para realizar el fine-tuning





# EJEMPLOS

MUCHAS GRACIAS  
POR SU ATENCIÓN