

## Prompt utilizado en Claude

Actúa como un experto en Python. Soy Juan Pablo Espina. Estoy realizando un curso sobre Generación de Prompts.

Mi entrega final tiene como consigna: "Identifica una problemática y plantea una solución concreta utilizando OpenIA o herramientas alternativas. Para resolver dicha problemática, deberás generar prompts a partir de los dos modelos vistos durante el curso: texto-texto y texto-imagen, justificando su uso y la resolución del problema. Desarrollarás una POC (proof of concept) que permita, a través de una jupyter notebook, mostrar una implementación utilizando las técnicas de Fast prompting para solucionar el problema seleccionado".

Los objetivos del trabajo son "1) Comprender cómo se genera un aviso. 2) Explorar estrategias para la creación de avisos efectivos. 3) Asegurar la viabilidad del proyecto, definiendo claramente su alcance y considerando limitaciones y recursos disponibles".

Yo seleccioné crear un chatbot en Whatsapp sobre preguntas frecuentes que realizan los estudiantes de la Facultad de Filosofía, Historia, Letras y Estudios Orientales, de la Universidad del Salvador.

Utilizaré como base para las preguntas frecuentes el archivo que adjunto en formato pdf.

Necesito que analices el archivo que adjunto con las preguntas frecuentes.

Necesito que programes el chatbot en Whatsapp sobre preguntas frecuentes en python y que me entregues el código para copiar y pegar en Google Colab.

Si necesitas alguna información extra, consultame. No inventes.

Necesito que me expliques el código que me vas a entregar en Python.

Como ejemplo, te paso el código de un chatbot de traducción al español en telegram, que vimos en clase. `pip install openai==0.28` `import openai` `def traducir_ingles_español(texto_ingles):` `""" Traduce texto del inglés al español usando OpenAI`

`Args: texto_ingles (str): El texto en inglés que se quiere traducir`

`Devuelve: str: El texto traducido al español """` `# Configurando la clave API` `openai.api_key = "sk-proj-P6fYZBkj75WkROM9Ca9IN81Slvq86POHpFEtd1KzRh9dfR8KI-H5twSXHz980UhQOmQNOCuEy4T3BIbkFJzKZIMJe_XKUxoO864lw5Jmjfcgl3AKoBDgbPDUAQI2026nWHnjRALU-ea4CSm-F18VTD4WRglA"` `try:` `# Realizar la petición a la API` `respuesta = openai.ChatCompletion.create(model="gpt-3.5-turbo", mensajes=[{"role": "system", "content": "Eres un traductor especializado en traducir de inglés a español."}, {"role": "user", "content": f"Traduce este texto al español: {texto_ingles}"}], temperatura=0.3 )`

`# Obtener la traducción` `traducción = respuesta.choices[0].message['content'].strip()` `return traducción`

`excepto Excepción como e:` `print(f"Error en la traducción: {str(e)}")` `return Ninguno`

`if name == " main ":` `# Puedes cambiar este texto por el que quieras traducir` `texto = "¡Hola! Este es un texto de muestra que queremos traducir al español."`

```
traduccion = traducir_ingles_español(texto)
```

```
if traduccion: print("\nTexto original (inglés):") print(texto) print("\nTraducción (español):")  
print(traduccion)
```

## Respuesta de Claude

Entendido. Desarrollaré un chatbot en Python para Whatsapp utilizando la API de OpenAI para responder preguntas frecuentes de la Facultad de Filosofía, Historia, Letras y Estudios Orientales.

Dado que necesitarás instalar algunas librerías, te proporcionaré un código completo para Google Colab.

```
# Instalación de librerías necesarias  
!pip install openai twilio  
  
import openai  
import json  
  
# Configuración de credenciales OpenAI  
openai.api_key = "TU_API_KEY_OPENAI"  
  
# Base de datos de preguntas frecuentes  
preguntas_frecuentes = {  
    "inscripcion_materias": """  
        Los alumnos deberán inscribirse a las materias a través del portal de  
servicios https://servicios.usal.edu.ar  
o de forma personal en la sede. La inscripción se realiza en marzo y  
julio.  
Requisitos:  
- Estar al día con aranceles  
- Haber presentado toda la documentación requerida  
- No tener materias con superposición horaria  
- Primer año: la inscripción la realiza la Secretaría Administrativa  
""",  
  
    "faltas_permitidas": """  
        Los alumnos deben cumplir con el 75% de asistencia.  
Cantidad de faltas según horas semanales:  
- 2 horas semanales (anual): 4 faltas  
- 3 horas semanales (anual): 6 faltas  
- 3 horas semanales (cuatrimestral): 3 faltas  
- 4 horas semanales (anual): 8 faltas  
- 5 horas semanales (anual): 10 faltas  
"""
```

```

- 6 horas semanales (anual): 12 faltas
"""

"prorroga_final": """
Para solicitar prórroga para rendir un final con escolaridad vencida:
1. Solicitar formulario en Secretaría Administrativa (abril)
2. Pedir al docente un tema de evaluación
3. Si apruebas la evaluación, quedas habilitado
4. Entregar formulario en Secretaría
Plazo máximo: 24 meses desde finalización del cursado
"""

"inscripcion_finales": """
Inscripción a exámenes finales:
- Por web en servicios.usal.edu.ar
- Hasta 72 hs hábiles antes del examen
- Requiere:
  * 75% de asistencia
  * Parciales aprobados
  * Materias correlativas aprobadas
  * Pagos al día
Turnos: Julio, Noviembre/Diciembre, Febrero/Marzo
"""

"final_sin_titulo_secundario": """
Para rendir en turno Julio sin título secundario:
1. Enviar mail a fhleo@usal.edu.ar
2. Asunto: Solicitar Excepción para rendir final
3. Adjuntar:
  - Formulario de solicitud
  - Foto/escaneo de certificado de título en trámite
Plazo límite: 18 de Junio
"""
}

def generar_respuesta_ia(pregunta):
    """
    Genera respuesta usando GPT para preguntas no cubiertas directamente
    """
    try:
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[
                {"role": "system", "content": "Eres un asistente de la Universidad del Salvador especializado en responder preguntas de estudiantes de la Facultad de Filosofía, Historia, Letras y Estudios Orientales."},

```

```

        {"role": "user", "content": pregunta}
    ],
    max_tokens=200,
    temperature=0.7
)
return response.choices[0].message['content'].strip()
except Exception as e:
    return f"Lo siento, hubo un error: {str(e)}"

def chatbot_usal(mensaje):
    """
    Función principal del chatbot
    """
    mensaje = mensaje.lower()

    # Mapeo de palabras clave a respuestas
    mapeo = {
        "materias": "inscripcion_materias",
        "inscripción": "inscripcion_materias",
        "faltas": "faltas_permitidas",
        "inasistencias": "faltas_permitidas",
        "prórroga": "prorroga_final",
        "final vencido": "prorroga_final",
        "exámenes": "inscripcion_finales",
        "finales": "inscripcion_finales",
        "título secundario": "final_sin_titulo_secundario"
    }

    # Buscar respuesta en mapeo de preguntas
    for palabra, categoria in mapeo.items():
        if palabra in mensaje:
            return preguntas_frecuentes[categoria]

    # Si no encuentra respuesta directa, usar IA
    return generar_respuesta_ia(mensaje)

# Ejemplo de uso
print(chatbot_usal("¿Cómo me inscribo a materias?"))
print(chatbot_usal("Cuántas faltas puedo tener"))

```

## Explicación del código:

### 1. Librerías:

- **openai**: Para generar respuestas con IA
- **Twilio** (comentado): Para integración con Whatsapp (implementación futura)

### 2. Estructura de datos:

- `preguntas_frecuentes`: Diccionario con respuestas predefinidas
- Cubre 5 categorías principales de preguntas
- 3. Funciones principales:
  - `generar_respuesta_ia()`: Usa GPT para respuestas no cubiertas
  - `chatbot_usa1()`: Función central que maneja la lógica del chatbot
    - Busca palabras clave
    - Devuelve respuestas predefinidas o usa IA

Pasos para usar:

1. Reemplazar `TU_API_KEY_OPENAI` con tu clave de OpenAI
2. Ejecutar en Google Colab
3. Probar con ejemplos de preguntas

Próximos pasos:

- Integración con Whatsapp (Twilio)
- Ampliar base de preguntas
- Mejorar la precisión de las respuestas

**Finalmente, le pregunté a Claude:** ¿Me explicás paso a paso cómo realizar la integración con Whatsapp (Twilio)?

### Respuesta de Claude

Te explicaré paso a paso cómo integrar el chatbot con WhatsApp usando Twilio.

# Instalación de librerías necesarias

```
!pip install flask twilio openai python-dotenv
```

```
from flask import Flask, request
from twilio.rest import Client
from twilio.twiml.messaging_response import MessagingResponse
import openai
import os
from dotenv import load_dotenv
```

```
# Cargar variables de entorno
load_dotenv()
```

```
# Configuración de credenciales
openai.api_key = os.getenv('OPENAI_API_KEY')
```

```

TWILIO_ACCOUNT_SID = os.getenv('TWILIO_ACCOUNT_SID')
TWILIO_AUTH_TOKEN = os.getenv('TWILIO_AUTH_TOKEN')
TWILIO_PHONE_NUMBER = os.getenv('TWILIO_PHONE_NUMBER')

# Inicializar Flask y Twilio
app = Flask(__name__)
client = Client(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN)

# Base de datos de preguntas frecuentes (mantener el diccionario anterior)
preguntas_frecuentes = {
    # ... (mantener el mismo diccionario de preguntas_frecuentes del código anterior)
}

def generar_respuesta_ia(pregunta):
    # ... (mantener la misma función del código anterior)
    pass

def chatbot_usal(mensaje):
    # ... (mantener la misma función del código anterior)
    pass

@app.route('/webhook', methods=['POST'])
def webhook():
    """
    Maneja los mensajes entrantes de WhatsApp
    """
    # Obtener el mensaje del usuario
    mensaje_entrante = request.values.get('Body', "").lower()
    numero_remitente = request.values.get('From', "")

    # Obtener respuesta del chatbot
    respuesta = chatbot_usal(mensaje_entrante)

    # Crear respuesta para WhatsApp
    response = MessagingResponse()
    response.message(respuesta)

    return str(response)

if __name__ == '__main__':
    app.run(debug=True)

```

## Explicación

Ahora te explico los pasos para implementar esta integración:

### 1. Crear cuenta en Twilio :

- Visita <https://www.twilio.com>
- Regístrese para obtener una cuenta gratuita
- Obtén tus credenciales: Account SID y Auth Token

### 2. Configurar WhatsApp Sandbox :

- En el panel de Twilio, ve a “Mensajería” → “Pruébalo” → “Enviar un mensaje de WhatsApp”
- Sigue las instrucciones para activar tu sandbox de WhatsApp
- Guarda el número de teléfono que te asignan

### 3. Crear archivo .env :

texto sin formato

Copiar

```
OPENAI_API_KEY=tu api key de openai
TWILIO_ACCOUNT_SID=tu account sid de twilio
TWILIO_AUTH_TOKEN=tu auth token de twilio
TWILIO_PHONE_NUMBER=tu numero de whatsapp de twilio
```

### 4. Configurar Webhook :

- Necesitas hacer tu servidor accesible desde Internet

Opciones: a) Usar ngrok (para desarrollo):

intento

Copiar

```
pip install ngrok
```

- `ngrok http 5000`

b) Desplegar en un servidor (para producción):

- HéroeKu
- Python en cualquier lugar
- Nube de Google
- etc.

### 5. Configurar URL del Webhook en Twilio :

- Veo el panel de Twilio
- En la configuración de WhatsApp Sandbox
- Agrega tu URL (ejemplo: <https://tu-dominio.com/webhook>)
- Seleccionar método POST

### 6. Probar el chatbot :

- Envía un mensaje al número de WhatsApp de tu sandbox
- El mensaje llegará a tu webhook
- El chatbot procesará y responderá automáticamente

Consideraciones importantes:

1. **Seguridad :**

- Nunca subas las credenciales directamente en el código
- Usa variables de entorno (.env)
- Implementar autenticación en el webhook

2. **Límites :**

- Cuenta gratuita de Twilio: limitada en mensajes/día
- Sandbox de WhatsApp: requiere que los usuarios se registren primero
- OpenAI: considera los límites de la API

3. **Mantenimiento :**

- Monitorea los errores
- Implementar el registro
- Considere implementar limitación de velocidad

4. **Costos :**

- Twilio: cobra por mensaje
- OpenAI: cobra por uso de API
- Alojamiento: depende del proveedor

Para producción, recomendaría:

1. Implementar manejo de errores robusto
2. Agregar sistema de registro
3. Limitación de velocidad de implementación
4. Agregar monitoreo
5. Considerar escalabilidad