# Bot de Atención al Cliente con RAG y GPT-4 – Código y Explicación Gráfico El contenido generado por IA puede ser incorrecto. CODE

*import os*

*import gradio as gr*

*import textwrap*

*from openai import OpenAI*

*from llama\_index.core import SimpleDirectoryReader, VectorStoreIndex*

*# ✅ OpenAI-Client initialisieren*

*client = OpenAI()*

*# ✅ Lokale Dateien laden (txt, csv, pdf)*

*documents = SimpleDirectoryReader(*

*input\_dir=".",*

*required\_exts=[".txt", ".csv", ".pdf"]*

*).load\_data()*

*# ✅ Vektorindex aus lokalen Dokumenten erstellen*

*index = VectorStoreIndex.from\_documents(documents)*

*query\_engine = index.as\_query\_engine()*

*# ✅ Chatfunktion: RAG + GPT-4 mit Systemrolle*

*def kundenfrage\_beantworten(frage):*

*try:*

*# ✳️ Antwort aus lokalem Index abrufen*

*rag\_response = query\_engine.query(frage).response*

*# ✳️ GPT-4 nutzt RAG-Antwort als Kontext, nicht als Frage*

*antwort = client.chat.completions.create(*

*model="gpt-4",*

*messages=[*

*{*

*"role": "system",*

*"content": (*

*"Du bist ein professioneller Assistent für Kundenservice. "*

*"Nutze den bereitgestellten Kontext, um die Nutzerfrage sachlich, freundlich und hilfreich zu beantworten. "*

*"Falls der Kontext keine Antwort erlaubt, gib bitte eine höfliche Standardantwort."*

*)*

*},*

*{*

*"role": "user",*

*"content": (*

*f"Kontext:\n{rag\_response}\n\n"*

*f"Frage: {frage}"*

*)*

*}*

*],*

*stream=False*

*)*

*# 🧾 Antworttext extrahieren & umbrechen*

*rohtext = antwort.choices[0].message.content.strip()*

*return textwrap.fill(rohtext, width=100)*

*except Exception as e:*

*return f"⚠️ Fehler: {str(e)}"*

*# ✅ Gradio-Interface starten*

*gr.Interface(*

*fn=kundenfrage\_beantworten,*

*inputs=gr.Textbox(label="Deine Frage", placeholder="Was möchten Sie wissen?"),*

*outputs=gr.TextArea(label="Antwort", lines=8, max\_lines=20),*

*title="<h3 style='font-size:18px'>🧠📁 Automatischer Kundenservice mit RAG & GPT-4</h3>",*

*theme="default"*

*).launch()*

**Explicación**

*import os*  
Importa el módulo os para rutas de archivos o variables de entorno (no se utiliza activamente aquí)

*import gradio as gr*Carga Gradio: el framework para la interfaz de usuario en el navegador

*import textwrap*Permite dividir automáticamente el texto en líneas de longitud fija

*from openai import OpenAI*Conecta tu script con la API de OpenAI (por ejemplo, GPT-4).

*from llama\_index.core import SimpleDirectoryReader, VectorStoreIndex*Importa componentes de llama-index para leer e indexar archivos locales

*client = OpenAI()*Inicializa el cliente de OpenAI con tu clave API (desde variables de entorno)

*documents = SimpleDirectoryReader(input\_dir=".", required\_exts=[".txt", ".csv", ".pdf"]).load\_data()*Lee los archivos locales (.txt, .csv, .pdf) desde la carpeta actual.

*index = VectorStoreIndex.from\_documents(documents)*Crea un índice vectorial a partir de los documentos.

*query\_engine = index.as\_query\_engine()*Convierte el índice en un motor de consultas para responder preguntas.

*def responder\_pregunta\_cliente(pregunta):*Función que se activa al recibir una pregunta del usuario.

*try:*Inicio del manejo de errores: evita que el Bot se bloquee.

*rag\_response = query\_engine.query(pregunta).response*Busca la parte más relevante de los documentos según la pregunta.

*respuesta = client.chat.completions.create(...)*Llama a GPT-4 y le pasa el rol, el contexto y la pregunta.

*model="gpt-4",*Especifica que se debe utilizar el modelo GPT-4.

*messages=[...]*Lista de mensajes: rol del sistema + pregunta con contexto.

*{"role": "system", "content": (...)}*Define que GPT debe actuar como asistente profesional de atención al cliente.

*{"role": "user", "content": (f"Contexto:\n{rag\_response}\n\nPregunta: {pregunta}")}*Mensaje real: contexto extraído + pregunta del usuario.

*stream=False*La respuesta se genera completamente antes de mostrarse.

*rohtext = antwort.choices[0].message.content.strip()*Extrae el texto de la respuesta de GPT y elimina espacios innecesarios.

*return textwrap.fill(texto\_crudo, width=100)*Formatea el texto en líneas de máximo 100 caracteres.

*except Exception as e:*En caso de error...

*return f"⚠️ Error: {str(e)}"*...muestra un mensaje de error amigable.

*gr.Interface(...)*Inicia la interfaz gráfica de Gradio.

*fn=responder\_pregunta\_cliente,*Asocia la función de respuesta con la entrada del usuario.

*inputs=gr.Textbox(...)*Campo de entrada donde el usuario escribe su pregunta.

*outputs=gr.TextArea(...)*Campo de salida donde aparece la respuesta con formato.

*title="..."*Título superior en la interfaz – con HTML y emojis.

*theme="default").launch()*Lanza la aplicación en el navegador con el tema visual por defecto.