### whatsapp\_store\_bot/main.py

from flask import Flask

from app.whatsapp\_webhook import whatsapp\_blueprint

app = Flask(\_\_name\_\_)

app.register\_blueprint(whatsapp\_blueprint)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000, debug=True)

### whatsapp\_store\_bot/app/whatsapp\_webhook.py

from flask import Blueprint, request, jsonify

from .openai\_agent import generate\_answer

from .pdf\_loader import extract\_text\_from\_pdf

from .whatsapp\_utils import enviar\_mensaje

whatsapp\_blueprint = Blueprint("whatsapp", \_\_name\_\_)

contexto = extract\_text\_from\_pdf("data/productos\_catalogo.pdf")

@whatsapp\_blueprint.route("/webhook", methods=["POST"])

def whatsapp\_webhook():

data = request.get\_json()

try:

mensaje = data["entry"][0]["changes"][0]["value"]["messages"][0]["text"]["body"]

numero = data["entry"][0]["changes"][0]["value"]["messages"][0]["from"]

respuesta = generate\_answer(mensaje, contexto)

enviar\_mensaje(numero, respuesta)

return jsonify({"status": "success"}), 200

except Exception as e:

return jsonify({"error": str(e)}), 400

### whatsapp\_store\_bot/app/openai\_agent.py

import os

import openai

from dotenv import load\_dotenv

load\_dotenv()

openai.api\_key = os.getenv("OPENAI\_API\_KEY")

def generate\_answer(user\_question, context):

messages = [

{"role": "system", "content": "Eres un asistente experto en una tienda virtual. Usa el contexto para responder."},

{"role": "user", "content": f"{context}\n\nPregunta: {user\_question}"}

]

response = openai.ChatCompletion.create(

model="gpt-4",

messages=messages

)

return response.choices[0].message["content"]

### whatsapp\_store\_bot/app/pdf\_loader.py

import fitz # PyMuPDF

def extract\_text\_from\_pdf(pdf\_path):

text = ""

with fitz.open(pdf\_path) as doc:

for page in doc:

text += page.get\_text()

return text

### whatsapp\_store\_bot/app/whatsapp\_utils.py

import os

import requests

from dotenv import load\_dotenv

load\_dotenv()

def enviar\_mensaje(numero, texto):

url = f"https://graph.facebook.com/v19.0/{os.getenv('WHATSAPP\_PHONE\_ID')}/messages"

headers = {

"Authorization": f"Bearer {os.getenv('WHATSAPP\_TOKEN')}",

"Content-Type": "application/json"

}

payload = {

"messaging\_product": "whatsapp",

"to": numero,

"type": "text",

"text": {"body": texto}

}

response = requests.post(url, headers=headers, json=payload)

print("Mensaje enviado:", response.status\_code, response.text)

### whatsapp\_store\_bot/.env (no compartir este archivo en público)

OPENAI\_API\_KEY=sk-...

WHATSAPP\_TOKEN=EAAG...

WHATSAPP\_PHONE\_ID=1234567890

### whatsapp\_store\_bot/requirements.txt

flask

openai

python-dotenv

requests

PyMuPDF

### whatsapp\_store\_bot/README.md

# Bot de WhatsApp para tienda virtual con OpenAI

Este bot responde automáticamente a preguntas de clientes sobre productos, usando WhatsApp y la API de OpenAI. El conocimiento del bot se entrena con un archivo PDF del catálogo de productos.

## Requisitos

- Cuenta en OpenAI con clave API

- Cuenta de WhatsApp Business (Meta) con número y token

- Archivo PDF con el catálogo de productos en `/data`

## Cómo ejecutar

```bash

python -m venv venv

source venv/bin/activate # Windows: .\venv\Scripts\activate

pip install -r requirements.txt

flask run --port=5000

```

Exponer con ngrok o localtunnel para pruebas:

```bash

ngrok http 5000

```

## Estructura

- `main.py`: arranca el servidor Flask

- `app/`: lógica principal del bot

- `.env`: credenciales (no compartir)

- `data/`: archivos PDF del catálogo

## Webhook WhatsApp

Registra tu webhook en Meta usando la URL pública de `ngrok` y configúralo con el método POST.