**Tecnologías a usar:**

| **Componente** | **Tecnología** |
| --- | --- |
| Interfaz WhatsApp | Twilio o 360dialog |
| Procesamiento del lenguaje | Python + spaCy o GPT API |
| Backend | Flask o FastAPI |
| Base de datos | SQLite / PostgreSQL / Firebase |
| Almacenamiento de voz | Whisper API o Google STT |
| Hospedaje | Render / Heroku / VPS |

**✅ Entregables que voy a generarte:**

1. Estructura completa del código en carpetas.
2. Archivos .py funcionales por módulo.
3. Script de instalación (requisitos).
4. Plantilla de base de datos.
5. Instrucciones para conectarlo a WhatsApp.
6. Documentación para desplegarlo.

¿Quieres que empecemos ahora por el **Módulo 1: conexión con WhatsApp y procesamiento básico de mensajes**?

Puedo entregarte el código inicial con Twilio y Flask para que ya comience a recibir y responder mensajes. ¿Te parece bien?

**Dijiste:**

si

**ChatGPT dijo:**

Easyshop Whatsapp Bot

Responde en el chat en su lugar

Ya tienes el **Módulo 1: Integración básica de WhatsApp con Python (Twilio + Flask)**. Este código permite:

* Recibir mensajes por WhatsApp.
* Procesar comandos básicos como "hola", "vendí" o "balance".
* Responder automáticamente.

**🧪 ¿Cómo probarlo?**

1. Crea cuenta en https://www.twilio.com/whatsapp.
2. Consigue un número sandbox.
3. Enlaza tu número de prueba.
4. Usa una URL pública como [ngrok](https://ngrok.com/) para exponer tu servidor Flask.
5. Configura el webhook con /whatsapp como endpoint.

# EASYSHOP - Módulo 4: Pedidos con aprobación por WhatsApp

from flask import Flask, request

from twilio.twiml.messaging\_response import MessagingResponse

import sqlite3

import re

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

# Inicializar base de datos y tablas adicionales para pedidos

def init\_db():

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute('''CREATE TABLE IF NOT EXISTS transacciones (

id INTEGER PRIMARY KEY AUTOINCREMENT,

tipo TEXT,

producto TEXT,

cantidad INTEGER,

precio\_unitario REAL,

total REAL,

fecha DATETIME DEFAULT CURRENT\_TIMESTAMP

)''')

c.execute('''CREATE TABLE IF NOT EXISTS inventario (

producto TEXT PRIMARY KEY,

cantidad INTEGER

)''')

c.execute('''CREATE TABLE IF NOT EXISTS pedidos (

id INTEGER PRIMARY KEY AUTOINCREMENT,

cliente TEXT,

estado TEXT, -- pendiente, aprobado, rechazado

total REAL,

fecha DATETIME DEFAULT CURRENT\_TIMESTAMP

)''')

c.execute('''CREATE TABLE IF NOT EXISTS detalle\_pedido (

id INTEGER PRIMARY KEY AUTOINCREMENT,

pedido\_id INTEGER,

producto TEXT,

cantidad INTEGER,

precio\_unitario REAL,

FOREIGN KEY(pedido\_id) REFERENCES pedidos(id)

)''')

conn.commit()

conn.close()

# Función para obtener precio promedio por producto

def obtener\_precio\_promedio(producto):

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("SELECT AVG(precio\_unitario) FROM transacciones WHERE producto = ?", (producto,))

res = c.fetchone()[0]

conn.close()

return res if res else 0

# Registrar pedido y detalles, devuelve id del pedido

def registrar\_pedido(cliente, productos):

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

total\_pedido = 0

# Insertar pedido pendiente con total 0 provisional

c.execute("INSERT INTO pedidos (cliente, estado, total) VALUES (?, 'pendiente', 0)", (cliente,))

pedido\_id = c.lastrowid

# Insertar detalles y calcular total

for prod, cant in productos.items():

precio = obtener\_precio\_promedio(prod)

total\_pedido += precio \* cant

c.execute("INSERT INTO detalle\_pedido (pedido\_id, producto, cantidad, precio\_unitario) VALUES (?, ?, ?, ?)",

(pedido\_id, prod, cant, precio))

# Actualizar total del pedido

c.execute("UPDATE pedidos SET total = ? WHERE id = ?", (total\_pedido, pedido\_id))

conn.commit()

conn.close()

return pedido\_id, total\_pedido

# Actualizar estado pedido y manejar inventario

def actualizar\_estado\_pedido(pedido\_id, nuevo\_estado):

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("UPDATE pedidos SET estado = ? WHERE id = ?", (nuevo\_estado, pedido\_id))

if nuevo\_estado == 'aprobado':

# Descontar stock

c.execute("SELECT producto, cantidad FROM detalle\_pedido WHERE pedido\_id = ?", (pedido\_id,))

detalles = c.fetchall()

for producto, cantidad in detalles:

c.execute("UPDATE inventario SET cantidad = cantidad - ? WHERE producto = ?", (cantidad, producto))

conn.commit()

conn.close()

# Consultar detalles de pedido

def obtener\_detalles\_pedido(pedido\_id):

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("SELECT producto, cantidad, precio\_unitario FROM detalle\_pedido WHERE pedido\_id = ?", (pedido\_id,))

resultados = c.fetchall()

conn.close()

return resultados

# Parsear texto de pedido tipo "pedido 2 arroz, 1 leche"

def parsear\_pedido(texto):

texto = texto.lower().replace("pedido", "").strip()

items = texto.split(",")

productos = {}

for item in items:

item = item.strip()

m = re.match(r"(\d+)\s+(\w+)", item)

if m:

cantidad = int(m.group(1))

producto = m.group(2)

productos[producto] = productos.get(producto, 0) + cantidad

return productos

@app.route("/whatsapp", methods=["POST"])

def whatsapp\_reply():

incoming\_msg = request.values.get('Body', '').lower()

from\_number = request.values.get('From')

response = MessagingResponse()

msg = response.message()

# Pedidos

if incoming\_msg.startswith("pedido"):

productos = parsear\_pedido(incoming\_msg)

if not productos:

msg.body("Formato incorrecto para pedido. Ejemplo: 'pedido 2 arroz, 1 leche'")

return str(response)

pedido\_id, total = registrar\_pedido(from\_number, productos)

msg.body(f"Pedido recibido con ID {pedido\_id}. Total aproximado: ${total:.2f}.\nPara aprobar: 'aprobar pedido {pedido\_id}'. Para rechazar: 'rechazar pedido {pedido\_id}'.")

return str(response)

# Aprobación/Rechazo de pedido

m\_approve = re.match(r"(aprobar|rechazar) pedido (\d+)", incoming\_msg)

if m\_approve:

accion = m\_approve.group(1)

pedido\_id = int(m\_approve.group(2))

actualizar\_estado\_pedido(pedido\_id, 'aprobado' if accion == 'aprobar' else 'rechazado')

estado\_texto = "aprobado" if accion == 'aprobar' else "rechazado"

msg.body(f"Pedido {pedido\_id} ha sido {estado\_texto}.")

return str(response)

# Aquí puedes mantener las otras funciones previas: saludo, balance, stock, ventas, gastos...

if "hola" in incoming\_msg:

msg.body("Hola, soy EASYSHOP. Puedes enviar pedidos con: 'pedido 2 arroz, 1 leche', o consultar balance, stock y ventas.")

elif "balance" in incoming\_msg:

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("SELECT SUM(total) FROM transacciones WHERE tipo='venta'")

ventas = c.fetchone()[0] or 0

c.execute("SELECT SUM(total) FROM transacciones WHERE tipo='gasto'")

gastos = c.fetchone()[0] or 0

conn.close()

ganancia = ventas - gastos

msg.body(f"Ventas: ${ventas:.0f}, Gastos: ${gastos:.0f}, Ganancia: ${ganancia:.0f}")

elif "stock" in incoming\_msg:

m = re.search(r"stock\s+(\w+)", incoming\_msg)

if m:

producto = m.group(1)

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("SELECT cantidad FROM inventario WHERE producto = ?", (producto,))

res = c.fetchone()

conn.close()

cantidad = res[0] if res else 0

msg.body(f"Stock disponible de {producto}: {cantidad}")

else:

msg.body("Especifica un producto. Ejemplo: 'stock arroz'")

elif "vendi" in incoming\_msg or "gasté" in incoming\_msg or "gaste" in incoming\_msg:

tipo = "venta" if "vendi" in incoming\_msg else "gasto"

m = re.search(r"\d+\s+(\w+)\s+a\s+(\d+)", incoming\_msg)

if m:

cantidad = int(re.search(r"\d+", m.group(0)).group())

producto = m.group(1)

precio = float(m.group(2))

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("INSERT INTO transacciones (tipo, producto, cantidad, precio\_unitario, total) VALUES (?, ?, ?, ?, ?)",

(tipo, producto, cantidad, precio, cantidad\*precio))

if tipo == "venta":

c.execute("UPDATE inventario SET cantidad = cantidad - ? WHERE producto = ?", (cantidad, producto))

else:

c.execute("INSERT INTO inventario (producto, cantidad) VALUES (?, ?) ON CONFLICT(producto) DO UPDATE SET cantidad = cantidad + ?",

(producto, cantidad, cantidad))

conn.commit()

conn.close()

msg.body(f"{tipo.title()} registrada: {cantidad} {producto} a ${precio:.0f} c/u")

else:

msg.body("Formato incorrecto. Usa: 'vendí 2 arroz a 1500'")

else:

msg.body("No entendí tu mensaje. Puedes enviar pedidos, consultar stock, balance, ventas, o saludar.")

return str(response)

if \_\_name\_\_ == "\_\_main\_\_":

init\_db()

app.run(debug=True)

Funcionalidades:

from flask import Flask, request

from twilio.twiml.messaging\_response import MessagingResponse

import sqlite3

import re

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

# Inicializar base de datos y tablas adicionales para pedidos

def init\_db():

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute('''CREATE TABLE IF NOT EXISTS transacciones (

id INTEGER PRIMARY KEY AUTOINCREMENT,

tipo TEXT,

producto TEXT,

cantidad INTEGER,

precio\_unitario REAL,

total REAL,

fecha DATETIME DEFAULT CURRENT\_TIMESTAMP

)''')

c.execute('''CREATE TABLE IF NOT EXISTS inventario (

producto TEXT PRIMARY KEY,

cantidad INTEGER

)''')

c.execute('''CREATE TABLE IF NOT EXISTS pedidos (

id INTEGER PRIMARY KEY AUTOINCREMENT,

cliente TEXT,

estado TEXT,

total REAL,

fecha DATETIME DEFAULT CURRENT\_TIMESTAMP

)''')

c.execute('''CREATE TABLE IF NOT EXISTS detalle\_pedido (

id INTEGER PRIMARY KEY AUTOINCREMENT,

pedido\_id INTEGER,

producto TEXT,

cantidad INTEGER,

precio\_unitario REAL,

FOREIGN KEY(pedido\_id) REFERENCES pedidos(id)

)''')

conn.commit()

conn.close()

# Función para obtener precio promedio por producto

def obtener\_precio\_promedio(producto):

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("SELECT AVG(precio\_unitario) FROM transacciones WHERE producto = ?", (producto,))

res = c.fetchone()[0]

conn.close()

return res if res else 0

# Registrar pedido y detalles, devuelve id del pedido

def registrar\_pedido(cliente, productos):

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

total\_pedido = 0

c.execute("INSERT INTO pedidos (cliente, estado, total) VALUES (?, 'pendiente', 0)", (cliente,))

pedido\_id = c.lastrowid

for prod, cant in productos.items():

precio = obtener\_precio\_promedio(prod)

total\_pedido += precio \* cant

c.execute("INSERT INTO detalle\_pedido (pedido\_id, producto, cantidad, precio\_unitario) VALUES (?, ?, ?, ?)",

(pedido\_id, prod, cant, precio))

c.execute("UPDATE pedidos SET total = ? WHERE id = ?", (total\_pedido, pedido\_id))

conn.commit()

conn.close()

return pedido\_id, total\_pedido

# Actualizar estado pedido y manejar inventario

def actualizar\_estado\_pedido(pedido\_id, nuevo\_estado):

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("UPDATE pedidos SET estado = ? WHERE id = ?", (nuevo\_estado, pedido\_id))

if nuevo\_estado == 'aprobado':

c.execute("SELECT producto, cantidad FROM detalle\_pedido WHERE pedido\_id = ?", (pedido\_id,))

detalles = c.fetchall()

for producto, cantidad in detalles:

c.execute("UPDATE inventario SET cantidad = cantidad - ? WHERE producto = ?", (cantidad, producto))

conn.commit()

conn.close()

# Consultar detalles de pedido

def obtener\_detalles\_pedido(pedido\_id):

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("SELECT producto, cantidad, precio\_unitario FROM detalle\_pedido WHERE pedido\_id = ?", (pedido\_id,))

resultados = c.fetchall()

conn.close()

return resultados

# Parsear texto de pedido tipo "pedido 2 arroz, 1 leche"

def parsear\_pedido(texto):

texto = texto.lower().replace("pedido", "").strip()

items = texto.split(",")

productos = {}

for item in items:

item = item.strip()

m = re.match(r"(\d+)\s+(\w+)", item)

if m:

cantidad = int(m.group(1))

producto = m.group(2)

productos[producto] = productos.get(producto, 0) + cantidad

return productos

@app.route("/whatsapp", methods=["POST"])

def whatsapp\_reply():

incoming\_msg = request.values.get('Body', '').lower()

from\_number = request.values.get('From')

response = MessagingResponse()

msg = response.message()

if incoming\_msg.startswith("pedido"):

productos = parsear\_pedido(incoming\_msg)

if not productos:

msg.body("Formato incorrecto para pedido. Ejemplo: 'pedido 2 arroz, 1 leche'")

return str(response)

pedido\_id, total = registrar\_pedido(from\_number, productos)

msg.body(f"Pedido recibido con ID {pedido\_id}. Total aproximado: ${total:.2f}.\nPara aprobar: 'aprobar pedido {pedido\_id}'. Para rechazar: 'rechazar pedido {pedido\_id}'.")

return str(response)

m\_approve = re.match(r"(aprobar|rechazar) pedido (\d+)", incoming\_msg)

if m\_approve:

accion = m\_approve.group(1)

pedido\_id = int(m\_approve.group(2))

actualizar\_estado\_pedido(pedido\_id, 'aprobado' if accion == 'aprobar' else 'rechazado')

estado\_texto = "aprobado" if accion == 'aprobar' else "rechazado"

msg.body(f"Pedido {pedido\_id} ha sido {estado\_texto}.")

return str(response)

if "hola" in incoming\_msg:

msg.body("Hola, soy EASYSHOP. Puedes enviar pedidos con: 'pedido 2 arroz, 1 leche', o consultar balance, stock y ventas.")

elif "balance" in incoming\_msg:

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("SELECT SUM(total) FROM transacciones WHERE tipo='venta'")

ventas = c.fetchone()[0] or 0

c.execute("SELECT SUM(total) FROM transacciones WHERE tipo='gasto'")

gastos = c.fetchone()[0] or 0

conn.close()

ganancia = ventas - gastos

msg.body(f"Ventas: ${ventas:.0f}, Gastos: ${gastos:.0f}, Ganancia: ${ganancia:.0f}")

elif "stock" in incoming\_msg:

m = re.search(r"stock\s+(\w+)", incoming\_msg)

if m:

producto = m.group(1)

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("SELECT cantidad FROM inventario WHERE producto = ?", (producto,))

res = c.fetchone()

conn.close()

cantidad = res[0] if res else 0

msg.body(f"Stock disponible de {producto}: {cantidad}")

else:

msg.body("Especifica un producto. Ejemplo: 'stock arroz'")

elif "vendi" in incoming\_msg or "gasté" in incoming\_msg or "gaste" in incoming\_msg:

tipo = "venta" if "vendi" in incoming\_msg else "gasto"

m = re.search(r"\d+\s+(\w+)\s+a\s+(\d+)", incoming\_msg)

if m:

cantidad = int(re.search(r"\d+", m.group(0)).group())

producto = m.group(1)

precio = float(m.group(2))

conn = sqlite3.connect('easyshop.db')

c = conn.cursor()

c.execute("INSERT INTO transacciones (tipo, producto, cantidad, precio\_unitario, total) VALUES (?, ?, ?, ?, ?)",

(tipo, producto, cantidad, precio, cantidad\*precio))

if tipo == "venta":

c.execute("UPDATE inventario SET cantidad = cantida

Paso 1: Preparar el entorno local para Python y Flask

Instala Python 3.8+

Descarga e instala Python desde https://python.org si no lo tienes.

Crear y activar un entorno virtual (opcional pero recomendado)

En tu consola (CMD/Terminal):

bash

Copiar

Editar

python -m venv easyshop-env

Windows:

bash

Copiar

Editar

easyshop-env\Scripts\activate

macOS/Linux:

bash

Copiar

Editar

source easyshop-env/bin/activate

Instalar dependencias

Ejecuta:

bash

Copiar

Editar

pip install flask twilio

Paso 2: Guardar el código

Crea un archivo llamado app.py en una carpeta de proyecto.

Copia y pega el código Python que te di (la app Flask completa).

Paso 3: Ejecutar la app localmente

En la consola (y con entorno activado):

bash

Copiar

Editar

python app.py

Esto iniciará un servidor local en http://127.0.0.1:5000

Paso 4: Exponer tu servidor local con Ngrok (para que Twilio acceda)

Descarga Ngrok https://ngrok.com/download

Ejecuta:

bash

Copiar

Editar

ngrok http 5000

Ngrok te mostrará una URL pública algo así:

https://xxxxxx.ngrok.io

Paso 5: Configurar Twilio para WhatsApp

Crea una cuenta en https://twilio.com y consigue un número de WhatsApp sandbox.

Ve a tu consola Twilio > Messaging > Try it Out > Send WhatsApp messages.

Sigue los pasos para conectar tu WhatsApp al Sandbox (enviar mensaje con código a un número).

En la configuración del sandbox, busca el campo "WHEN A MESSAGE COMES IN" y pon la URL de tu ngrok más la ruta /whatsapp, por ejemplo:

arduino

Copiar

Editar

https://xxxxxx.ngrok.io/whatsapp

Guarda la configuración.

Paso 6: Probar la app

Ahora, envía un mensaje de WhatsApp al número sandbox configurado, por ejemplo:

hola

vendí 3 arroz a 1500

pedido 2 arroz, 1 leche

balance

stock arroz

Y deberías recibir respuestas automáticas desde tu app.

Nota adicional: Variables de entorno para Twilio

El código que te di usa Flask y Twilio para recibir mensajes, pero para enviar mensajes con Twilio API (por ejemplo para confirmaciones automáticas fuera del webhook) podrías necesitar configurar account\_sid y auth\_token. Esto no es obligatorio para el webhook básico.