

Tudhor

- Julian David Tintinago
- Maria Camila Ramirez
- Santiago Acosta

si, ayudame a crear el data set para un chatbot para atención a multiples usuarios a partir de estas preguntas y genere las respuestas a cada pregunta como si fueras un ingeniero encargado del servicio al cliente y que tiene que dar soluciones rapidez y eficientes:

1. ¿costos de mantenimiento en todo lo referente a hardware?
2. ¿Qué disponibilidad de horarios de atención de ingenieros hay para el mantenimiento?
3. ¿control de sistema operativo en todos los ámbitos?
4. ¿cómo actualizo el sistema operativo?
5. ¿Hay alguna actualización disponible para mi sistema?
6. ¿mi aplicación no esta funcionando correctamente, podrías ayudarme a solucionarlo y guiarme paso a paso?
7. ¿Cuál es el estado actual del sistema o servicio?
8. ¿Cómo reinicio/reseteo mi aplicación o dispositivo?
9. ¿Cómo puedo crear una guía de usuario interactiva para mi aplicación, sistema o página web, que sea fácil de entender para usuarios no técnicos?
10. ¿Qué herramientas o plataformas recomiendas para crear un sistema de tickets de soporte automatizado con respuestas predefinidas?
11. ¿Cómo puedo configurar alertas automáticas para monitorear el rendimiento del sistema?
12. ¿Qué medidas de control debo implementar para gestionar los permisos de usuario?

esto para diseñar e implementar un chatbot inteligente que evolucione desde un asistente básico hasta un sistema adaptativo con análisis y trazabilidad usando el modelo regresión lineal a través de las doce preguntas que te dimos

| Categoría | Pregunta | Respuesta | Palabras clave |
|-----------|---|---|---|
| Servicios | ¿Qué tipo de soluciones a medida ofrecen exactamente? | Ofrecemos soluciones personalizadas en software, hardware, automatización industrial, inteligencia artificial y mantenimiento, diseñadas según las necesidades específicas de cada cliente. | soluciones personalizadas, software, hardware, automatización, inteligencia artificial, mantenimiento |

| | | | |
|-----------|---|---|---|
| Servicios | ¿Desarrollan software personalizado desde cero o adaptan soluciones existentes? | Ambas opciones. Podemos desarrollar software desde cero o adaptar plataformas ya existentes para ajustarlas a los requerimientos técnicos y operativos de cada empresa. | software personalizado, desarrollo, adaptación, soluciones existentes |
| Servicios | ¿También fabrican o ensamblan hardware propio? | Sí. Diseñamos, fabricamos y ensamblamos hardware especializado para procesos automatizados, controladores industriales y dispositivos embebidos según el proyecto. | hardware, diseño, fabricación, ensamblaje, controladores industriales |
| Servicios | ¿Qué tipos de sistemas de automatización industrial implementan? | Implementamos sistemas como PLC, SCADA, HMI, control de procesos, monitoreo remoto, robótica industrial y soluciones integradas con sensores y actuadores inteligentes. | automatización industrial, PLC, SCADA, HMI, robótica, sensores |
| Servicios | ¿Pueden integrar IA en procesos ya existentes? | Sí. Analizamos la infraestructura actual para incorporar algoritmos de inteligencia artificial que mejoran la toma de decisiones, el control de calidad, la predicción de fallas y la eficiencia operativa. | IA, procesos existentes, eficiencia, calidad, predicción de fallas |

| | | | |
|---------------------|---|---|---|
| Servicios | ¿Realizan mantenimiento preventivo o solo correctivo? | Ofrecemos mantenimiento preventivo, correctivo y predictivo. Nuestro objetivo es minimizar paros no programados y extender la vida útil de los equipos. | mantenimiento preventivo, correctivo, predictivo, vida útil |
| Servicios | ¿Qué incluye su servicio de mantenimiento ? | Incluye diagnóstico técnico, reparación, calibración, sustitución de partes, programación, actualizaciones de software, limpieza especializada y soporte técnico permanente. | mantenimiento, diagnóstico, reparación, soporte, programación |
| IA y Automatización | ¿Cómo puede la IA ayudar a mejorar mi línea de producción? | La IA permite identificar cuellos de botella, predecir fallas antes de que ocurran, ajustar parámetros en tiempo real, optimizar recursos y reducir desperdicios. | IA, producción, eficiencia, optimización, fallas |
| IA y Automatización | ¿Qué tipo de datos necesitan para implementar soluciones de IA? | Datos históricos de producción, variables del proceso, registros de mantenimiento, sensores en tiempo real y cualquier otro dato estructurado o no estructurado relacionado con la operación. | datos, IA, sensores, registros, producción |

| | | | |
|-----------------------------|---|---|--|
| IA y Automatización | ¿Cuáles son los beneficios reales de automatizar mis procesos? | Mayor precisión, reducción de errores humanos, aumento en la velocidad de producción, mejor control de calidad, reducción de costos operativos y mejor trazabilidad. | automatización, precisión, calidad, reducción de costos |
| IA y Automatización | ¿La automatización puede aplicarse en pequeños negocios o solo en industrias grandes? | Puede aplicarse en ambos. Ofrecemos soluciones escalables que se adaptan al tamaño y nivel tecnológico de cada empresa. | automatización, PYMES, escalabilidad, soluciones adaptadas |
| IA y Automatización | ¿Qué seguridad ofrecen los sistemas automatizados ? | Todos nuestros sistemas siguen normas de seguridad industrial y ciberseguridad. Implementamos sensores, paradas de emergencia, sistemas redundantes y monitoreo remoto. | seguridad, automatización, ciberseguridad, monitoreo, normas |
| Proyectos y Personalización | ¿Qué tan personalizado puede ser un proyecto? | Completamente. Desde la ingeniería, diseño, programación, interfaz, integración de hardware y soporte, nos adaptamos a las necesidades de cada cliente. | personalización, diseño, programación, soporte, ingeniería |

| | | | |
|-----------------------------|---|---|---|
| Proyectos y Personalización | ¿Pueden adaptar una solución que ya tengo instalada? | Sí. Evaluamos las soluciones actuales para mejorarlas, ampliarlas o integrarlas con nuevas tecnologías. | adaptación, solución existente, integración tecnológica |
| Proyectos y Personalización | ¿Cuánto tiempo tarda el desarrollo de un proyecto típico? | Depende del alcance. Proyectos pequeños pueden tardar entre 2 y 4 semanas, mientras que proyectos complejos pueden requerir de 2 a 3 meses o más. | desarrollo, duración, cronograma, proyecto |
| Proyectos y Personalización | ¿Qué tan involucrado debo estar durante el proceso de desarrollo? | Promovemos una participación activa del cliente en todas las etapas clave para asegurar que la solución cumpla con sus expectativas. | participación, cliente, colaboración, desarrollo |
| Proyectos y Personalización | ¿Ofrecen pruebas piloto o prototipos? | Sí. Podemos desarrollar prototipos o versiones piloto para validar el funcionamiento de la solución antes de su implementación definitiva. | prototipo, piloto, validación, pruebas |
| Costos y Financiamiento | ¿Cómo se calculan los costos de un proyecto? | Según el alcance técnico, tiempo estimado de desarrollo, cantidad de recursos requeridos, licencias y complejidad de la integración. | costos, presupuesto, desarrollo, alcance |

| | | | |
|-------------------------|--|--|--|
| Costos y Financiamiento | ¿Ofrecen planes de financiamiento o pagos por fases? | Sí. Contamos con opciones flexibles de pago por etapas según el avance del proyecto y los acuerdos comerciales con el cliente. | financiamiento, pagos, fases, flexibilidad |
| Costos y Financiamiento | ¿Tienen precios especiales para empresas del Eje Cafetero? | Sí. Como empresa local, ofrecemos tarifas preferenciales y planes especiales para impulsar la industria regional. | precios especiales, Eje Cafetero, descuentos |
| Integración y Soporte | ¿Sus sistemas son compatibles con los que ya uso? | Sí. Evaluamos sus sistemas actuales y adaptamos nuestras soluciones para lograr una integración eficiente sin reemplazos innecesarios. | compatibilidad, sistemas existentes, integración |
| Integración y Soporte | ¿Qué tipo de soporte ofrecen después de implementar la solución? | Brindamos soporte técnico continuo, mantenimiento programado, actualizaciones de software y atención a incidentes según el contrato de servicio. | soporte, mantenimiento, actualizaciones, postventa |
| Integración y Soporte | ¿Ofrecen capacitación para el personal? | Sí. Ofrecemos capacitación técnica al personal involucrado para garantizar un uso adecuado y eficiente del sistema. | capacitación, formación, entrenamiento, personal |

| | | | |
|-----------------------|---|--|--|
| Integración y Soporte | ¿Tienen atención 24/7 en caso de fallas? | Contamos con servicio técnico 24/7 para clientes con contratos activos que requieren soporte crítico o atención inmediata. | soporte 24/7, emergencias, servicio técnico |
| Integración y Soporte | ¿Qué pasa si una solución falla o no cumple las expectativas? | Ofrecemos garantías, ajustes, actualizaciones o reembolsos parciales, según el caso, siempre buscando la satisfacción del cliente. | garantía, solución fallida, satisfacción, reembolso |
| Impacto y Beneficios | ¿Qué mejoras puedo esperar en eficiencia o reducción de costos? | Clientes reportan mejoras de entre 15% y 40% en productividad, reducción de desperdicios y menor tiempo de inactividad. | eficiencia, reducción de costos, productividad |
| Impacto y Beneficios | ¿Tienen casos de éxito que puedan compartir? | Sí. Contamos con estudios de caso en agroindustria, manufactura y logística que muestran el impacto de nuestras soluciones. | casos de éxito, resultados, agroindustria, manufactura |
| Impacto y Beneficios | ¿Cuál es el retorno de inversión (ROI) esperado con sus soluciones? | En promedio, el ROI se alcanza entre los 6 y 18 meses posteriores a la implementación, dependiendo del sector. | ROI, retorno de inversión, rentabilidad, tiempo |
| Impacto y Beneficios | ¿Qué industrias han beneficiado más de sus servicios? | Trabajamos con empresas del sector agroindustrial, manufactura, logística, salud, comercio y educación técnica. | industrias, sectores, agroindustria, manufactura |

| | | | |
|-----------------------|--|--|---|
| Ubicación y Cobertura | ¿Trabajan solo en el Eje Cafetero o también en otras regiones? | Nuestra base está en el Eje Cafetero, pero atendemos proyectos en todo el territorio nacional. | cobertura, Eje Cafetero, otras regiones, nacional |
| Ubicación y Cobertura | ¿Pueden hacer visitas técnicas a mi planta o negocio? | Sí. Realizamos visitas técnicas para diagnóstico, implementación, seguimiento o mantenimiento según se requiera. | visitas técnicas, diagnóstico, mantenimiento |
| Ubicación y Cobertura | ¿Cómo es el proceso para solicitar una cotización? | Puedes contactarnos por correo, teléfono o formulario web. Agendamos una reunión de diagnóstico inicial y luego entregamos una cotización detallada. | cotización, contacto, proceso, reunión |

1.Código inicial

```
!pip install python-telegram-bot nest_asyncio openpyxl

import pandas as pd
import nest_asyncio
import asyncio
from telegram.ext import ApplicationBuilder, MessageHandler,
ContextTypes, filters
from telegram import Update
from datetime import datetime
import os

nest_asyncio.apply()

# Dataset inicial (puede ser reemplazado por uno nuevo)
df = pd.DataFrame({
    "Pregunta": [
        "costos de mantenimiento en todo lo referente a hardware",
        "disponibilidad de horarios de atención de ingenieros",
        "control de sistema operativo",
        "cómo actualizo el sistema operativo",
```



```

        "hay alguna actualización disponible para mi sistema",
        "mi aplicación no está funcionando correctamente",
        "estado actual del sistema o servicio",
        "cómo reinicio mi aplicación o dispositivo",
        "crear guía de usuario interactiva",
        "herramientas para sistema de tickets automatizado",
        "configurar alertas automáticas de rendimiento",
        "gestionar permisos de usuario"
    ],
    "Respuesta": [
        "Los costos dependen del hardware específico. Podemos ofrecerte una cotización si nos indicas el equipo exacto.",
        "Nuestros ingenieros están disponibles de lunes a viernes de 8am a 6pm, con guardias los fines de semana.",
        "Tenemos control total del sistema operativo: versiones, configuraciones y usuarios.",
        "Puedes actualizar el sistema operativo desde Configuración > Sistema > Actualizaciones.",
        "Puedes verificar actualizaciones en Configuración > Sistema. Si hay disponibles, aparecerán allí.",
        "Claro, dime qué error ves y te guiaré paso a paso para solucionarlo.",
        "Actualmente el sistema está funcionando con normalidad. ¿Quieres que revise algo específico?",
        "Para reiniciar, mantén presionado el botón de encendido o ve a Configuración > Reiniciar.",
        "Recomiendo usar herramientas como Guidely, Intro.js o Scribe. ¿Quieres que te muestre un ejemplo?",
        "Puedes usar Freshdesk, Zendesk o Zoho Desk para configurar respuestas predefinidas en tickets.",
        "Te sugiero usar Zabbix, Datadog o Prometheus para configurar alertas automáticas.",
        "Para gestionar permisos, crea roles y asigna privilegios según nivel. Usa un sistema RBAC."
    ]
})

# Archivo de registro
LOG_FILE = "registro_interacciones_chatbot.xlsx"

# Lista de usuarios saludados
usuarios_saludados = set()

```

```

# Palabras clave de despedida
palabras_despedida = ["adiós", "gracias", "hasta luego", "nos vemos",
"chao", "bye"]

# ✅ Función para actualizar el dataset desde archivo externo
def cargar_dataset_nuevo(ruta_archivo):
    global df
    if ruta_archivo.endswith(".csv"):
        df = pd.read_csv(ruta_archivo)
        print("✅ Dataset actualizado desde CSV.")
    elif ruta_archivo.endswith(".xlsx") or
ruta_archivo.endswith(".xls"):
        df = pd.read_excel(ruta_archivo)
        print("✅ Dataset actualizado desde Excel.")
    else:
        print("⚠️ Formato no soportado. Usa .csv o .xlsx.")

# Función de búsqueda de respuesta
def buscar_respuesta(pregunta_usuario):
    for i, pregunta in enumerate(df["Pregunta"]):
        if pregunta.lower() in pregunta_usuario.lower():
            return df["Respuesta"][i]
    return "Lo siento, no encontré una respuesta directa. ¿Puedes reformular tu pregunta?"

# Manejo del mensaje
async def handle_message(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    user_message = update.message.text.lower()
    username = update.effective_user.username or
update.effective_user.first_name
    respuesta = ""

    # Mensaje de bienvenida si es el primer mensaje
    if username not in usuarios_saludados:
        respuesta += f";Hola {username}! 😊 Bienvenido/a al centro de
soporte. Estoy aquí para ayudarte.\n\n"
        usuarios_saludados.add(username)

    # Mensaje de despedida si detecta palabras clave
    if any(palabra in user_message for palabra in palabras_despedida):
        respuesta += ";Gracias por comunicarte con nosotros! Si
necesitas algo más, no dudes en escribirnos. 🙌"

```

```

else:
    respuesta += buscar_respuesta(user_message)

# Enviar respuesta
await update.message.reply_text(respuesta)

# Guardar interacción
log = {
    "Usuario": username,
    "Pregunta": update.message.text,
    "Respuesta": respuesta,
    "Fecha": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
}

try:
    historial = pd.read_excel(LOG_FILE)
    historial = pd.concat([historial, pd.DataFrame([log])],
ignore_index=True)
except FileNotFoundError:
    historial = pd.DataFrame([log])
    historial.to_excel(LOG_FILE, index=False)

```

```

# Actualiza el dataset en caliente desde archivo
cargar_dataset_nuevo("Preguntas_INGE_LEAN_SAS.xlsx") # o
"nuevo_dataset.csv"

```

```

# Reemplaza con el token de tu bot
TOKEN = "8341715432:AAGiaMU9uC1ouieKl2P7FCPlOSDQ-hIaFPI"

app = ApplicationBuilder().token(TOKEN).build()
app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_message))

print("🤖 Bot activo en Telegram. ¡Habla con él!")
await app.run_polling()

```

```

import time

try:
    print("🕒 Manteniendo Colab activo...")
    while True:
        time.sleep(15)
except KeyboardInterrupt:

```

```
print("🛑 Bot detenido.")
```

2.codigo actualizado con machine learning

📦 Paso 1: Instalar librerías necesarias

```
!pip install transformers datasets torch scikit-learn openpyxl  
python-telegram-bot nest_asyncio
```

📖 Paso 2: Importar librerías

```
import pandas as pd  
import torch  
import random  
import nest_asyncio  
import asyncio  
from sklearn.preprocessing import LabelEncoder  
from datasets import Dataset  
from transformers import DistilBertTokenizerFast,  
DistilBertForSequenceClassification, Trainer, TrainingArguments  
from telegram.ext import ApplicationBuilder, MessageHandler,  
ContextTypes, filters  
from telegram import Update  
from datetime import datetime
```

```
nest_asyncio.apply()
```

📁 Paso 3: Subir archivo de preguntas

```
def subir_archivo():  
    from google.colab import files  
    uploaded = files.upload()  
    return next(iter(uploaded))
```

```
filename = subir_archivo()  
df = pd.read_excel(filename)  
df = df.dropna(subset=["Pregunta", "Categoría", "Respuesta"])  
df["Pregunta"] = df["Pregunta"].str.lower()
```

🧠 Paso 4: Preparar los datos

```
label_encoder = LabelEncoder()  
df["label"] = label_encoder.fit_transform(df["Categoría"])  
  
tokenizer =  
DistilBertTokenizerFast.from_pretrained("distilbert-base-uncased")
```

```

dataset = Dataset.from_pandas(df[["Pregunta", "label"]])

def tokenize_function(example):
    return tokenizer(example["Pregunta"], padding="max_length",
truncation=True)

tokenized_dataset = dataset.map(tokenize_function, batched=True)

# 🧠 Paso 5: Cargar y entrenar modelo
model = DistilBertForSequenceClassification.from_pretrained(
    "distilbert-base-uncased",
    num_labels=len(label_encoder.classes_)
)

training_args = TrainingArguments(
    output_dir="./results",
    per_device_train_batch_size=8,
    num_train_epochs=3,
    logging_steps=10,
    save_steps=50,
    save_total_limit=1,
    learning_rate=2e-5,
    weight_decay=0.01,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    tokenizer=tokenizer
)

trainer.train()

# 📖 Diccionario de respuestas
respuestas_dict =
df.groupby("Categoría")["Respuesta"].apply(list).to_dict()

# 🔍 Función de predicción
def predecir_intencion_transformers(pregunta):
    inputs = tokenizer(pregunta.lower(), return_tensors="pt",
truncation=True, padding=True)
    outputs = model(**inputs)

```

```

prediction = torch.argmax(outputs.logits, dim=1).item()
return label_encoder.inverse_transform([prediction])[0]

# 🤖 Telegram Bot
LOG_FILE = "registro_interacciones_chatbot_transformers.xlsx"
usuarios_saludados = set()
palabras_despedida = ["adiós", "gracias", "hasta luego", "nos vemos",
"chao", "bye"]

# 📌 Función principal del bot
async def handle_message(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    user_message = update.message.text.lower()
    username = update.effective_user.username or
update.effective_user.first_name
    respuesta = ""

    if username not in usuarios_saludados:
        respuesta += f";Hola {username}! 😊 Bienvenido/a al centro de
soporte. Estoy aquí para ayudarte.\n\n"
        usuarios_saludados.add(username)

    if any(palabra in user_message for palabra in palabras_despedida):
        respuesta += ";Gracias por comunicarte con nosotros! Si
necesitas algo más, no dudes en escribirnos. 🙌"
    else:
        try:
            intencion = predecir_intencion_transformers(user_message)
            posibles_respuestas = respuestas_dict.get(intencion, [])
            respuesta += random.choice(posibles_respuestas) if
posibles_respuestas else "Estoy entrenando para darte mejores
respuestas pronto. 😊"
        except Exception:
            respuesta += "Lo siento, no entendí tu pregunta. ¿Puedes
reformularla?"

    await update.message.reply_text(respuesta)

log = {
    "Usuario": username,
    "Pregunta": update.message.text,
    "Respuesta": respuesta,
    "Fecha": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
}

```

```

    }

    try:
        historial = pd.read_excel(LOG_FILE)
        historial = pd.concat([historial, pd.DataFrame([log])],
ignore_index=True)
    except FileNotFoundError:
        historial = pd.DataFrame([log])
        historial.to_excel(LOG_FILE, index=False)

# 🚀 Lanzar el bot
TOKEN = "8341715432:AAGiaMU9uC1ouieKl2P7FCPl0SDQ-hIaFPI" # 🔑
Reemplaza esto por tu token
app = ApplicationBuilder().token(TOKEN).build()
app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_message))

print("🤖 Bot activo en Telegram. ¡Habla con él!")
await app.run_polling()

```

3.código actualizado con múltiples usuarios, reinicio de chatbot por palabras clave y finalización, guardado de api en W & B donde se guardan sesiones iniciadas, archivo cargable de dataset en excel, cambio de nombre e imagen de la empresa.

```

# 📦 Paso 1: Instalar librerías necesarias
!pip install transformers datasets torch scikit-learn openpyxl
python-telegram-bot nest_asyncio

# 📖 Paso 2: Importar librerías
import pandas as pd
import torch
import random
import nest_asyncio
import asyncio
from sklearn.preprocessing import LabelEncoder
from datasets import Dataset
from transformers import DistilBertTokenizerFast,
DistilBertForSequenceClassification, Trainer, TrainingArguments
from telegram.ext import ApplicationBuilder, MessageHandler,
ContextTypes, filters
from telegram import Update
from datetime import datetime, timedelta

```

```

nest_asyncio.apply()

# 🕒 Registro de última interacción
ultimo_mensaje_usuario = {}

# 📁 Paso 3: Subir archivo de preguntas
def subir_archivo():
    from google.colab import files
    uploaded = files.upload()
    return next(iter(uploaded))

filename = subir_archivo()
df = pd.read_excel(filename)
df = df.dropna(subset=["Pregunta", "Categoría", "Respuesta"])
df["Pregunta"] = df["Pregunta"].str.lower()

# 🧠 Paso 4: Preparar los datos
label_encoder = LabelEncoder()
df["label"] = label_encoder.fit_transform(df["Categoría"])

tokenizer =
DistilBertTokenizerFast.from_pretrained("distilbert-base-uncased")
dataset = Dataset.from_pandas(df[["Pregunta", "label"]])

def tokenize_function(example):
    return tokenizer(example["Pregunta"], padding="max_length",
truncation=True)

tokenized_dataset = dataset.map(tokenize_function, batched=True)

# 🧠 Paso 5: Cargar y entrenar modelo
model = DistilBertForSequenceClassification.from_pretrained(
    "distilbert-base-uncased",
    num_labels=len(label_encoder.classes_)
)

training_args = TrainingArguments(
    output_dir="./results",
    per_device_train_batch_size=8,
    num_train_epochs=3,
    logging_steps=10,
    save_steps=50,

```



```

        save_total_limit=1,
        learning_rate=2e-5,
        weight_decay=0.01,
    )

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    tokenizer=tokenizer
)

trainer.train()

# 📖 Diccionario de respuestas
respuestas_dict =
df.groupby("Categoría")["Respuesta"].apply(list).to_dict()

# 🔍 Función de predicción
def predecir_intencion_transformers(pregunta):
    inputs = tokenizer(pregunta.lower(), return_tensors="pt",
truncation=True, padding=True)
    outputs = model(**inputs)
    prediction = torch.argmax(outputs.logits, dim=1).item()
    return label_encoder.inverse_transform([prediction])[0]

# 🤖 Telegram Bot
LOG_FILE = "registro_interacciones_chatbot_transformers.xlsx"
usuarios_saludados = set()
usuarios_activos = {} # 🆕 Almacena el último tiempo activo por
usuario
palabras_despedida = ["adiós", "gracias", "hasta luego", "nos vemos",
"chao", "bye"]
palabras_reinicio = ["reiniciar", "empezar de nuevo", "volver a
empezar", "menú", "menu"]

# 🔄 Reinicio automático por inactividad
async def verificar_inactividad():
    while True:
        ahora = datetime.now()
        for user, ultimo in list(usuarios_activos.items()):
            if (ahora - ultimo) > timedelta(minutes=5):
                if user in usuarios_saludados:

```

```

        usuarios_saludados.remove(user)
        usuarios_activos.pop(user)
        await asyncio.sleep(60) # Verifica cada minuto

# 📌 Función principal del bot
async def handle_message(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    user_message = update.message.text.lower()
    username = update.effective_user.username or
update.effective_user.first_name
    respuesta = ""

    usuarios_activos[username] = datetime.now()

    # Reinicio manual del chat
    if any(p in user_message for p in palabras_reinicio):
        if username in usuarios_saludados:
            usuarios_saludados.remove(username)
            respuesta += f"🔄 Reiniciando el chat, {username}. ¡Hola de
nuevo! ¿En qué puedo ayudarte? 😊"
            await update.message.reply_text(respuesta)
            return

    # Mensaje de bienvenida si es la primera vez o después de
inactividad
    if username not in usuarios_saludados:
        respuesta += f"¡Hola {username}! 😊 Bienvenido/a al centro de
soporte. Estoy aquí para ayudarte.\n\n"
        usuarios_saludados.add(username)

    # Despedida
    if any(palabra in user_message for palabra in palabras_despedida):
        respuesta += "¡Gracias por comunicarte con nosotros! Que tengas
un excelente día. 🌟👋"
        if username in usuarios_saludados:
            usuarios_saludados.remove(username)
        await update.message.reply_text(respuesta)
        return

    # Respuesta automática
    try:
        intencion = predecir_intencion_transformers(user_message)
        posibles_respuestas = respuestas_dict.get(intencion, [])

```

```

        respuesta += random.choice(posibles_respuestas) if
posibles_respuestas else "Estoy entrenando para darte mejores
respuestas pronto. 😊"
    except Exception:
        respuesta += "Lo siento, no entendí tu pregunta. ¿Puedes
reformularla?"

    await update.message.reply_text(respuesta)

# Guardar en log
log = {
    "Usuario": username,
    "Pregunta": update.message.text,
    "Respuesta": respuesta,
    "Fecha": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
}

try:
    historial = pd.read_excel(LOG_FILE)
    historial = pd.concat([historial, pd.DataFrame([log])],
ignore_index=True)
except FileNotFoundError:
    historial = pd.DataFrame([log])
    historial.to_excel(LOG_FILE, index=False)

# 🚀 Lanzar el bot
TOKEN = "8341715432:AAGiaMU9uC1ouieKl2P7FCPlOSDQ-hIaFPI" # 🔑
Reemplaza esto por tu token
app = ApplicationBuilder().token(TOKEN).build()
app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_message))

# 🧠 Iniciar verificación de inactividad
async def iniciar_bot():
    print("🤖 Bot activo en Telegram. ¡Habla con él!")
    asyncio.create_task(verificar_inactividad()) # 🆕 Tarea paralela
    await app.run_polling()

await iniciar_bot()

```

4.codigo actualizado con despliegue de menú de servicios

```
# 📦 Paso 1: Instalar librerías necesarias
!pip install transformers datasets torch scikit-learn openpyxl
python-telegram-bot nest_asyncio

# 📖 Paso 2: Importar librerías
import pandas as pd
import torch
import random
import nest_asyncio
import asyncio

from sklearn.preprocessing import LabelEncoder
from datasets import Dataset
from transformers import DistilBertTokenizerFast,
DistilBertForSequenceClassification, Trainer, TrainingArguments
from telegram.ext import ApplicationBuilder, MessageHandler,
ContextTypes, filters
from telegram import Update
from datetime import datetime, timedelta

nest_asyncio.apply()

# 🕒 Registro de última interacción
ultimo_mensaje_usuario = {}

# 📁 Paso 3: Subir archivo de preguntas
def subir_archivo():
    from google.colab import files
    uploaded = files.upload()
    return next(iter(uploaded))

filename = subir_archivo()
df = pd.read_excel(filename)
df = df.dropna(subset=["Pregunta", "Categoría", "Respuesta"])
df["Pregunta"] = df["Pregunta"].str.lower()

# 🧠 Paso 4: Preparar los datos
label_encoder = LabelEncoder()
df["label"] = label_encoder.fit_transform(df["Categoría"])

tokenizer =
DistilBertTokenizerFast.from_pretrained("distilbert-base-uncased")
dataset = Dataset.from_pandas(df[["Pregunta", "label"]])
```

```

def tokenize_function(example):
    return tokenizer(example["Pregunta"], padding="max_length",
truncation=True)

tokenized_dataset = dataset.map(tokenize_function, batched=True)

# 🧠 Paso 5: Cargar y entrenar modelo
model = DistilBertForSequenceClassification.from_pretrained(
    "distilbert-base-uncased",
    num_labels=len(label_encoder.classes_)
)

training_args = TrainingArguments(
    output_dir="./results",
    per_device_train_batch_size=8,
    num_train_epochs=3,
    logging_steps=10,
    save_steps=50,
    save_total_limit=1,
    learning_rate=2e-5,
    weight_decay=0.01,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    tokenizer=tokenizer
)

trainer.train()

# 📖 Diccionario de respuestas
respuestas_dict =
df.groupby("Categoría")["Respuesta"].apply(list).to_dict()

# 🔍 Función de predicción
def predecir_intencion_transformers(pregunta):
    inputs = tokenizer(pregunta.lower(), return_tensors="pt",
truncation=True, padding=True)
    outputs = model(**inputs)
    prediction = torch.argmax(outputs.logits, dim=1).item()

```

```
return label_encoder.inverse_transform([prediction])[0]
```

 Menú principal de servicios

```
menu_principal = ""
```

 *MENÚ DE SERVICIOS DISPONIBLES*

 1 *Inteligencia Artificial (IA)*


- └ 1.1 Integración de IA en procesos ya existentes
- └ 1.2 Análisis de datos históricos y de sensores
- └ 1.3 Optimización de procesos (fallas, eficiencia)
- └ 1.4 Estudio de retorno de inversión (ROI)
- └ 1.5 Aplicación en agroindustria, manufactura, logística
- └ 1.6 IA para empresas grandes y PYMES
- └ 1.7 Machine Learning en procesos industriales
- └ 1.8 Análisis de datos con IA
- └ 1.9 Cloud computing para IA

 *Software*

- └ 2.1 Desarrollo de software desde cero
- └ 2.2 Adaptación e integración de software
- └ 2.3 Interfaces personalizadas y prototipos
- └ 2.4 Capacitación y compatibilidad con sistemas
- └ 2.5 Desarrollo de Apps móviles industriales
- └ 2.6 Programación y análisis en la nube
- └ 2.7 Sistemas de monitoreo y reportes (CSV, PDF, XLS)

 *Hardware*

- └ 3.1 Fabricación de hardware especializado
- └ 3.2 Dispositivos embebidos y dataloggers
- └ 3.3 Sensores para temperatura, humedad, etc.
- └ 3.4 Integración hardware-software
- └ 3.5 Desarrollo de tableros eléctricos
- └ 3.6 Levantamiento de planos eléctricos industriales

 *Automatización Industrial*

- └ 4.1 Sistemas PLC, SCADA, HMI
- └ 4.2 Automatización con sensores y robótica
- └ 4.3 Soluciones escalables para empresas
- └ 4.4 Seguridad industrial y control eléctrico
- └ 4.5 Integración M2M y telemetría
- └ 4.6 Monitoreo remoto y servidores de procesos

 *Mantenimiento*

```

└─ 5.1 Mantenimiento preventivo, correctivo y predictivo
└─ 5.2 Diagnóstico, reparación y calibración
└─ 5.3 Soporte técnico 24/7 y garantías
└─ 5.4 Capacitación del equipo operativo
└─ 5.5 Planes de pago y tarifas regionales
└─ 5.6 Gestión postventa y mejora continua
└─ 5.7 Telemetría para mantenimiento proactivo
"""

# 📄 Función para mostrar el menú
async def mostrar_menu(update, context, username):
    mensaje_bienvenida = f"¡Hola {username}! 😊 Bienvenido/a al centro de soporte. Estoy aquí para ayudarte.\n"
    await update.message.reply_text(mensaje_bienvenida)
    await update.message.reply_text(menu_principal,
    parse_mode="Markdown")

# 🤖 Telegram Bot
LOG_FILE = "registro_interacciones_chatbot_transformers.xlsx"
usuarios_saludados = set()
usuarios_activos = {}
palabras_despedida = ["adiós", "gracias", "hasta luego", "nos vemos", "chao", "bye"]
palabras_reinicio = ["reiniciar", "empezar de nuevo", "volver a empezar", "menú", "menu"]

# 🔄 Verificación de inactividad
async def verificar_inactividad():
    while True:
        ahora = datetime.now()
        for user, ultimo in list(usuarios_activos.items()):
            if (ahora - ultimo) > timedelta(minutes=5):
                if user in usuarios_saludados:
                    usuarios_saludados.remove(user)
                usuarios_activos.pop(user)
        await asyncio.sleep(60)

# 📌 Función principal del bot
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_message = update.message.text.lower()
    username = update.effective_user.username or
    update.effective_user.first_name

```

```

respuesta = ""

usuarios_activos[username] = datetime.now()

if any(p in user_message for p in palabras_reinicio):
    if username in usuarios_saludados:
        usuarios_saludados.remove(username)
    await mostrar_menu(update, context, username)
    return

if username not in usuarios_saludados:
    usuarios_saludados.add(username)
    await mostrar_menu(update, context, username)
    return

if any(palabra in user_message for palabra in palabras_despedida):
    respuesta += "¡Gracias por comunicarte con nosotros! Que tengas
un excelente día. 🌟👏"
    if username in usuarios_saludados:
        usuarios_saludados.remove(username)
    await update.message.reply_text(respuesta)
    return

try:
    intencion = predecir_intencion_transformers(user_message)
    posibles_respuestas = respuestas_dict.get(intencion, [])
    respuesta += random.choice(posibles_respuestas) if
posibles_respuestas else "Estoy entrenando para darte mejores
respuestas pronto. 😊"
except Exception:
    respuesta += "Lo siento, no entendí tu pregunta. ¿Puedes
reformularla?"

await update.message.reply_text(respuesta)

log = {
    "Usuario": username,
    "Pregunta": update.message.text,
    "Respuesta": respuesta,
    "Fecha": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
}

try:

```



```

        historial = pd.read_excel(LOG_FILE)
        historial = pd.concat([historial, pd.DataFrame([log])],
ignore_index=True)
    except FileNotFoundError:
        historial = pd.DataFrame([log])
    historial.to_excel(LOG_FILE, index=False)

# 🚀 Lanzar el bot
TOKEN = "8341715432:AAGiaMU9uC1ouieKl2P7FCPl0SDQ-hIaFPI" # Reemplázalo
por el tuyo
app = ApplicationBuilder().token(TOKEN).build()
app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_message))

# 🧠 Iniciar verificación de inactividad
async def iniciar_bot():
    print("🤖 Bot activo en Telegram. ¡Habla con él!")
    asyncio.create_task(verificar_inactividad())
    await app.run_polling()

await iniciar_bot()

```

5.codigo actualizado con mensaje final de más información en pagina web y soporte técnico continuo con carga de archivos excel para dataset

```

# 📄 Función para mostrar el menú
async def mostrar_menu(update, context, username):
    mensaje_bienvenida = f";Hola {username}! 😊 Bienvenido/a al centro
de soporte. Estoy aquí para ayudarte.\n"
    await update.message.reply_text(mensaje_bienvenida)
    await update.message.reply_text(menu_principal,
parse_mode="Markdown")

# 🤖 Telegram Bot
LOG_FILE = "registro_interacciones_chatbot_transformers.xlsx"
usuarios_saludados = set()
usuarios_activos = {}
palabras_despedida = ["adiós", "gracias", "hasta luego", "nos vemos",
"chao", "bye"]
palabras_reinicio = ["reiniciar", "empezar de nuevo", "volver a
empezar", "menú", "menu"]

```

```
# 🔄 Verificación de inactividad
```

```
async def verificar_inactividad():
```

```
    while True:
```

```
        ahora = datetime.now()
```

```
        for user, ultimo in list(usuarios_activos.items()):
```

```
            if (ahora - ultimo) > timedelta(minutes=5):
```

```
                if user in usuarios_saludados:
```

```
                    usuarios_saludados.remove(user)
```

```
                    usuarios_activos.pop(user)
```

```
            await asyncio.sleep(60)
```

```
# 📌 Función principal del bot
```

```
async def handle_message(update: Update, context:
```

```
ContextTypes.DEFAULT_TYPE):
```

```
    user_message = update.message.text.lower()
```

```
    username = update.effective_user.username or
```

```
update.effective_user.first_name
```

```
    respuesta = ""
```

```
    usuarios_activos[username] = datetime.now()
```

```
    if any(p in user_message for p in palabras_reinicio):
```

```
        if username in usuarios_saludados:
```

```
            usuarios_saludados.remove(username)
```

```
        await mostrar_menu(update, context, username)
```

```
        return
```

```
    if username not in usuarios_saludados:
```

```
        usuarios_saludados.add(username)
```

```
    await mostrar_menu(update, context, username)
```

```
    return
```

```
    if any(palabra in user_message for palabra in palabras_despedida):
```

```
        respuesta += "¡Gracias por comunicarte con nosotros! Que tengas  
un excelente día. 🌟👋, para mas información ingresa a nuestra pagina
```

```
web: https://ingelean.com/#service-page"
```

```
    if username in usuarios_saludados:
```

```
        usuarios_saludados.remove(username)
```

```

        await update.message.reply_text(respuesta)
        return

    try:
        intencion = predecir_intencion_transformers(user_message)
        posibles_respuestas = respuestas_dict.get(intencion, [])
        respuesta += random.choice(posibles_respuestas) if
posibles_respuestas else "Estoy entrenando para darte mejores
respuestas pronto. 😊"
    except Exception:
        respuesta += "Lo siento, no entendí tu pregunta. ¿Puedes
reformularla?"

    await update.message.reply_text(respuesta)

log = {
    "Usuario": username,
    "Pregunta": update.message.text,
    "Respuesta": respuesta,
    "Fecha": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
}

    try:
        historial = pd.read_excel(LOG_FILE)
        historial = pd.concat([historial, pd.DataFrame([log])],
ignore_index=True)
    except FileNotFoundError:
        historial = pd.DataFrame([log])
    historial.to_excel(LOG_FILE, index=False)

# 🚀 Lanzar el bot
TOKEN = "8341715432:AAGiaMU9uC1ouieKl2P7FCPl0SDQ-hIaFPI" # Reemplázalo
por el tuyo
app = ApplicationBuilder().token(TOKEN).build()
app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_message))

```

```
# 🧠 Iniciar verificación de inactividad
async def iniciar_bot():
    print("🤖 Bot activo en Telegram. ¡Habla con él!")
    asyncio.create_task(verificar_inactividad())
    await app.run_polling()

await iniciar_bot()
```