```
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup, InputFile

from telegram.ext import Application, CommandHandler, MessageHandler, CallbackQueryHandler, filters, CallbackContext

import random

import requests


TOKEN = '7252132869:AAH98TVVz-HePCnmgnRFpCI4HHhMgwL_5-Y'# clave del bot

WEATHER_API_KEY = '243c7ea889bdaa796511a0b45967c842'  # clave de OpenWeatherMap

PDF_FILE_PATH = 'D:\\sistemas_inteligentes\\botFinal\\ejercicioFinal.pdf'  # Ruta al archivo PDF

#1 estado de animo

estado_animo = {

   'feliz': "Que bueno que te sientas feliz",

   'triste': "Siento escuchar eso.",

   'enojado': "Espero que las cosas mejoren.",

   'neutral': "Gracias por compartir tu estado."

}

#2 trivia de animo

trivia_questions = {

   'feliz': [

      {"question": "¿Cuál es la capital de Francia?",

       "options": ['Paris', 'Londres', 'Madrid'],

       "answer": 'Paris'},

      {"question": "¿Quién pinto la Mona Lisa?",

       "options": ['Vincent van Gogh', 'Leonardo da Vinci', 'Pablo Picasso'],

       "answer": 'Leonardo da Vinci'}

   ],
```

```
  'triste': [
    {"question": "¿Cual es el planeta mas cercano al Sol?",
     "options": ['Venus', 'Marte', 'Mercurio'],
     "answer": 'Mercurio'},
    {"question": "¿Cuál es el río mas largo del mundo?",
     "options": ['Nilo', 'Amazonas', 'Juarez'],
     "answer": 'Nilo'}
  ],
  'enojado': [
    {"question": "¿Cuál es el metal más valioso del mundo?",
     "options": ['Oro', 'Plata', 'Platino'],
     "answer": 'Oro'},
    {"question": "¿Qué continente es conocido como la cuna de la civilización?",
     "options": ['Asia', 'África', 'Europa'],
     "answer": 'África'}
  ],
  'neutral': [
    {"question": "¿Cuál es la distancia de la Tierra al Sol?",
     "options": ['150 millones de kilómetros', '200 millones de kilómetros', '250 millones de kilómetros'],
     "answer": '150 millones de kilómetros'},
    {"question": "¿Qué animal es conocido como el rey de la selva?",
     "options": ['León', 'Tigre', 'Elefante'],
     "answer": 'León'}
  ]
}
#preguntas de cultura
cultura_questions = [
```

```
    {"question": "¿Quién escribió 'Don Quijote de la Mancha'?",
     "options": ['Miguel de Cervantes', 'William Shakespeare', 'Gabriel García Márquez'],
     "answer": 'Miguel de Cervantes'},
    {"question": "¿Cuál es el pais más grande del mundo?",
     "options": ['Rusia', 'Canada', 'China'],
     "answer": 'Rusia'},
    {"question": "¿En qué año llegó el hombre a la Luna?",
     "options": ['1965', '1969', '1972'],
     "answer": '1969'},
    {"question": "¿Cual es la capital de Japon?",
     "options": ['Hiroshima', 'Hokaido', 'Tokio'],
     "answer": 'Tokio'},
    {"question": "¿Cuál es el océano más grande?",
     "options": ['Atlántico', 'Indico', 'Pacifico'],
     "answer": 'Pacifico'}
]

# Información de productos
products = [
    {'name': 'Gorra', 'price': 100},
    {'name': 'Tenis', 'price': 500},
    {'name': 'Playera', 'price': 200},
    {'name': 'Patineta', 'price': 400},
    {'name': 'Audífonos', 'price': 500}
]

user_mood = {}
```

```
user_data = {}

cart = []

#clima

async def send_weather_info(update: Update, context: CallbackContext) -> None:

  location = update.message.text

  url = f'http://api.openweathermap.org/data/2.5/weather?q={location}&appid={WEATHER_API_KEY}&units=metric'

  response = requests.get(url).json()


  if response.get('cod') != 200:

    await update.message.reply_text("No se pudo obtener la información del clima. Verifica la ubicación e intenta nuevamente.")

    return


  city = response['name']

  temp = response['main']['temp']

  weather_description = response['weather'][0]['description']

  await update.message.reply_text(f"Clima en {city}:\nTemperatura: {temp}°C\nDescripción: {weather_description.capitalize()}")

#trivia

async def send_trivia_question(query, context: CallbackContext, mood: str) -> None:

  questions = trivia_questions.get(mood, [])

  if not questions:

    await query.message.reply_text("No hay preguntas de trivia")

    return


  question = random.choice(questions)
```

```
    keyboard = [[InlineKeyboardButton(option,
callback_data=f"answer_{mood}_{option}")] for option in question['options']]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await query.message.reply_text(question['question'],
reply_markup=reply_markup)


#preguntas de cultura
async def send_cultura_question(query, context: CallbackContext, index: int) ->
None:
    question = cultura_questions[index]
    keyboard = [[InlineKeyboardButton(option,
callback_data=f"cultura_{index}_{option}")] for option in question['options']]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await query.message.reply_text(question['question'],
reply_markup=reply_markup)


#inicio
async def inicio(update: Update, context: CallbackContext) -> None:
    if update.message.text.lower() == "inicio":
        keyboard = [
            [InlineKeyboardButton("Opción 1", callback_data='1')],
            [InlineKeyboardButton("Opción 2", callback_data='2')],
            [InlineKeyboardButton("Opción 3", callback_data='3')],
            [InlineKeyboardButton("Opción 4", callback_data='4')],
            [InlineKeyboardButton("Opción 5", callback_data='5')],
            [InlineKeyboardButton("Opción 6", callback_data='6')],
        ]
        reply_markup = InlineKeyboardMarkup(keyboard)
        await update.message.reply_text("Elige una opción:",
reply_markup=reply_markup)
```

```
async def button(update: Update, context: CallbackContext) -> None:
    query = update.callback_query
    await query.answer()
    data = query.data


    if data == '1':
        keyboard = [
            [InlineKeyboardButton("Estado de Ánimo", callback_data='1_1')],
            [InlineKeyboardButton("Trivia", callback_data='1_2')],
            [InlineKeyboardButton("Volver al Menú Principal", callback_data='start')]
        ]
        reply_markup = InlineKeyboardMarkup(keyboard)
        await query.edit_message_text(text='Opción 1:', reply_markup=reply_markup)
    elif data == '1_1':
        keyboard = [
            [InlineKeyboardButton("Feliz", callback_data='estado_feliz')],
            [InlineKeyboardButton("Triste", callback_data='estado_triste')],
            [InlineKeyboardButton("Enojado", callback_data='estado_enojado')],
            [InlineKeyboardButton("Neutral", callback_data='estado_neutral')],
            [InlineKeyboardButton("Volver al Menú Principal", callback_data='start')]
        ]
        reply_markup = InlineKeyboardMarkup(keyboard)
        await query.edit_message_text(text='Selecciona tu estado de animo:',
reply_markup=reply_markup)
    elif data == '1_2':
        if 'mood' in user_mood:
            mood = user_mood['mood']
```

```
            await send_trivia_question(query, context, mood)
        else:
            await query.edit_message_text(text='Selecciona tu estado de animo primero.')
    elif data.startswith('estado_'):
        mood = data.split('_')[1]
        response = estado_animo.get(mood, 'Estado de animo no reconocido.')
        user_mood['mood'] = mood
        await query.edit_message_text(text=response + "\n elige 'Trivia' para responder una pregunta.")
    elif data.startswith('answer_'):
        _, mood, answer = data.split('_')
        questions = trivia_questions.get(mood, [])
        question = next((q for q in questions if answer in q['options']), None)
        correct_answer = question['answer'] if question else None
        result = "¡Correcto! 🎉" if answer == correct_answer else f"Incorrecto. La respuesta correcta es: {correct_answer}."
        await query.edit_message_text(text=result)
        user_mood.pop('mood', None)  # Resetea el estado de ánimo para la próxima interacción
    elif data == '2':
        keyboard = [
            [InlineKeyboardButton("Informacion Personal", callback_data='2_1')],
            [InlineKeyboardButton("Preguntas de Cultura", callback_data='2_2')],
            [InlineKeyboardButton("Volver al Menú Principal", callback_data='start')]
        ]
        reply_markup = InlineKeyboardMarkup(keyboard)
        await query.edit_message_text(text='Opción 2:', reply_markup=reply_markup)
    elif data == '2_1':
```

```
        await query.edit_message_text(text='¿Cual es tu nombre?')
        user_data['step'] = 'name'
    elif data == '2_2':
        await send_cultura_question(query, context, 0)
        user_data['cultura_step'] = 0
    elif data.startswith('cultura_'):
        index, answer = map(int, data.split('_')[1:])
        question = cultura_questions[index]
        correct_answer = question['answer']
        if answer == question['options'].index(correct_answer):
            if index + 1 < len(cultura_questions):
                await send_cultura_question(query, context, index + 1)
                user_data['cultura_step'] = index + 1
            else:
                await query.edit_message_text(text="¡Correcto! 🎉 Has terminado todas las preguntas.")
                user_data.pop('cultura_step', None)
        else:
            await query.edit_message_text(text=f"Incorrecto. La respuesta correcta es: {correct_answer}. Inténtalo de nuevo.")
            await send_cultura_question(query, context, index)
    elif data == '3':
        keyboard = [
            [InlineKeyboardButton("Ver Productos", callback_data='3_1')],
            [InlineKeyboardButton("Ventas de Productos", callback_data='3_2')],
            [InlineKeyboardButton("Volver al Menú Principal", callback_data='start')]
        ]
        reply_markup = InlineKeyboardMarkup(keyboard)
        await query.edit_message_text(text='Opción 3:', reply_markup=reply_markup)
```

```
    elif data == '3_1':
        keyboard = [[InlineKeyboardButton(p['name'], callback_data=f"product_{i}")] for i, p in enumerate(products)]
        reply_markup = InlineKeyboardMarkup(keyboard)
        await query.edit_message_text(text='Selecciona un producto para añadir al carrito:', reply_markup=reply_markup)
    elif data.startswith('product_'):
        product_index = int(data.split('_')[1])
        cart.append(products[product_index])
        await query.edit_message_text(text=f"Producto {products[product_index]['name']} añadido al carrito.")
    elif data == '3_2':
        total = sum(item['price'] for item in cart)
        if cart:
            cart_summary = "\n".join([f"{item['name']}: ${item['price']}" for item in cart])
            await query.edit_message_text(text=f"Resumen de ventas:\n{cart_summary}\nTotal de ventas: ${total:.2f}")
        else:
            await query.edit_message_text(text="El carrito está vacío.")
    elif data == '4':
        keyboard = [
            [InlineKeyboardButton("Consultar Clima", callback_data='4_1')],
            [InlineKeyboardButton("Volver al Menú Principal", callback_data='start')]
        ]
        reply_markup = InlineKeyboardMarkup(keyboard)
        await query.edit_message_text(text='Opción 4:', reply_markup=reply_markup)
    elif data == '4_1':
        await query.edit_message_text(text="Escribe el nombre de la ciudad para obtener el clima.")
        user_data['step'] = 'weather'
```

```
    elif data == '5':
        keyboard = [
            [InlineKeyboardButton("Descargar PDF", callback_data='5_1')],
            [InlineKeyboardButton("Salir", callback_data='5_2')],
            [InlineKeyboardButton("Volver al Menú Principal", callback_data='start')]
        ]
        reply_markup = InlineKeyboardMarkup(keyboard)
        await query.edit_message_text(text='Opción 5:', reply_markup=reply_markup)
    elif data == '5_1':
        await query.message.reply_document(document=open(PDF_FILE_PATH, 'rb'))
    elif data == '5_2':
        await query.message.reply_text("¡Hasta luego!")
        await query.message.delete()  # Opcional: Eliminar el mensaje del bot al salir
    elif data == '6':
        await query.edit_message_text(text="Opción V6:  estar activado en la nube.")

async def handle_message(update: Update, context: CallbackContext) -> None:
    if 'step' in user_data:
        step = user_data['step']
        text = update.message.text
        if step == 'name':
            user_data['name'] = text
            await update.message.reply_text(f"Hola, {text}. ¿Cuál es tu edad?")
            user_data['step'] = 'age'
        elif step == 'age':
            user_data['age'] = text
            await update.message.reply_text(f"Tu edad es {text}. ¿Cuál es tu número de
teléfono?")
```

```
        user_data['step'] = 'phone'
    elif step == 'phone':
        user_data['phone'] = text
        await update.message.reply_text(f"Tu teléfono es {text}. ¿Cuál es tu dirección?")
        user_data['step'] = 'address'
    elif step == 'address':
        user_data['address'] = text
        await update.message.reply_text(
            f"Tu dirección es {text}.\nNombre: {user_data['name']}\nEdad: {user_data['age']}\nTeléfono: {user_data['phone']}\nDirección: {user_data['address']}"
        )
        user_data.clear()  # Resetea los datos de usuario para la próxima interacción
    elif step == 'weather':
        await send_weather_info(update, context)
        user_data.pop('step', None)  # Resetea el paso de consulta del clima
  else:
    await inicio(update, context)

async def start(update: Update, context: CallbackContext) -> None:
  await update.message.reply_text("Escribe 'inicio' para comenzar.")

def main() -> None:
  application = Application.builder().token(TOKEN).build()
  application.add_handler(CommandHandler('start', start))
  application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_message))
  application.add_handler(CallbackQueryHandler(button))
```

```
    application.run_polling()


if __name__ == '__main__':
    main()
```