EDUCACIÓN
SECRETARÍA DE EDUCACIÓN

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLOGICO DE PACHUCA

2.2

BOT DE TELEGRAM

INSTITUTO TECNOLÓGICO DE PACHUCA

NOMBRE DE LA CARRERA:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

GRUPO: B

MATERIA: LENGUAJES AUTOMATAS I
ING. BAUME LAZCANO RODOLFO

ALUMNA:
CITLALI MARTÍNEZ SÁNCHEZ 21200614

FECHA DE ENTREGA:
15 DE ABRIL DEL 2024

CÓDIGO

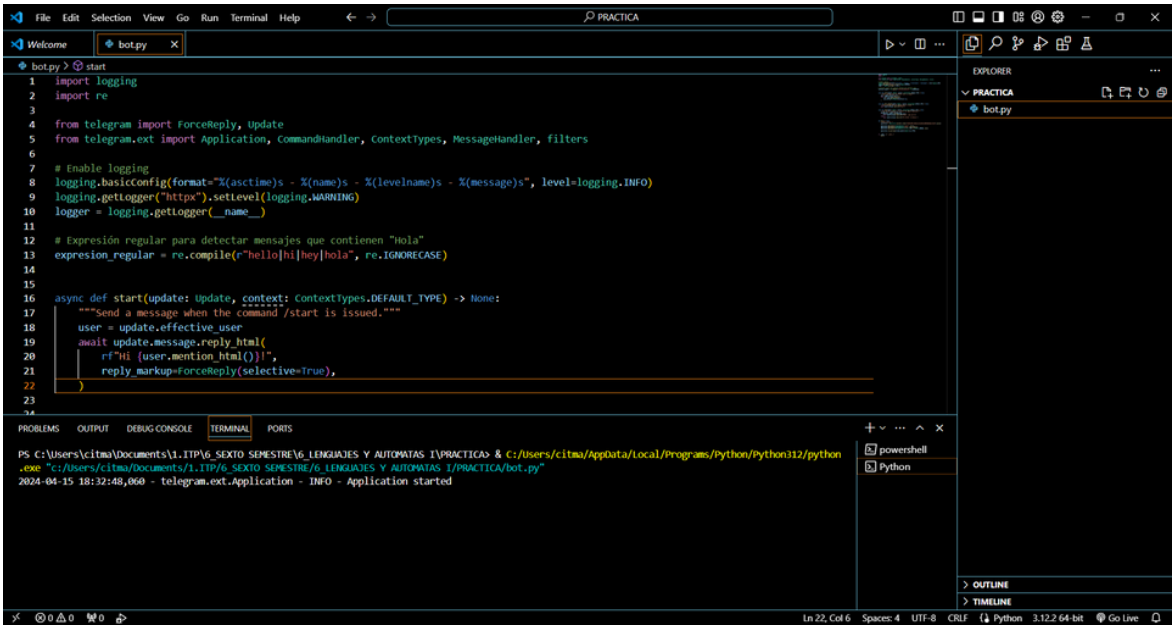
```
import logging
import re

from telegram import ForceReply, Update
from telegram.ext import Application, CommandHandler, ContextTypes, MessageHandler, filters


logging.basicConfig(format="%(asctime)s - %(name)s - %(levelname)s - %(message)s", level=logging.INFO)
logging.getLogger("httpx").setLevel(logging.WARNING)
logger = logging.getLogger(__name__)


expresion_regular_saludo = re.compile(r"hello|hi|hey|hola", re.IGNORECASE)

expresion_regular_personalizada = re.compile(r"bien|excelente|tu", re.IGNORECASE)

patron_origen_destino_fecha = r"volar de (\w+) a (\w+) el (\d{1,2} de \w+)"
patron_precio = r"cuánto cuesta un vuelo de (\w+) a (\w+)"
patron_ida_vuelta = r"un vuelo de ida y vuelta de (\w+) a (\w+)"


async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
  """Enviar un mensaje cuando se emite el comando /start."""
  user = update.effective_user
  await update.message.reply_html(
    rf"Hola {user.mention_html()}!",
    reply_markup=ForceReply(selective=True),
  )


async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
  """Enviar un mensaje cuando se emite el comando /help."""
  await update.message.reply_text("Help!")


async def echo(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
  """Repetir el mensaje del usuario si coincide con alguna de las expresiones regulares."""
  message_text = update.message.text
  if expresion_regular_saludo.search(message_text):
    await update.message.reply_text("¡Hola! ¿Cómo estás?")
  elif expresion_regular_personalizada.search(message_text):
    await update.message.reply_text("¡Me alegro!, me encuentro de maravilla.")
  else:
```

CÓDIGO

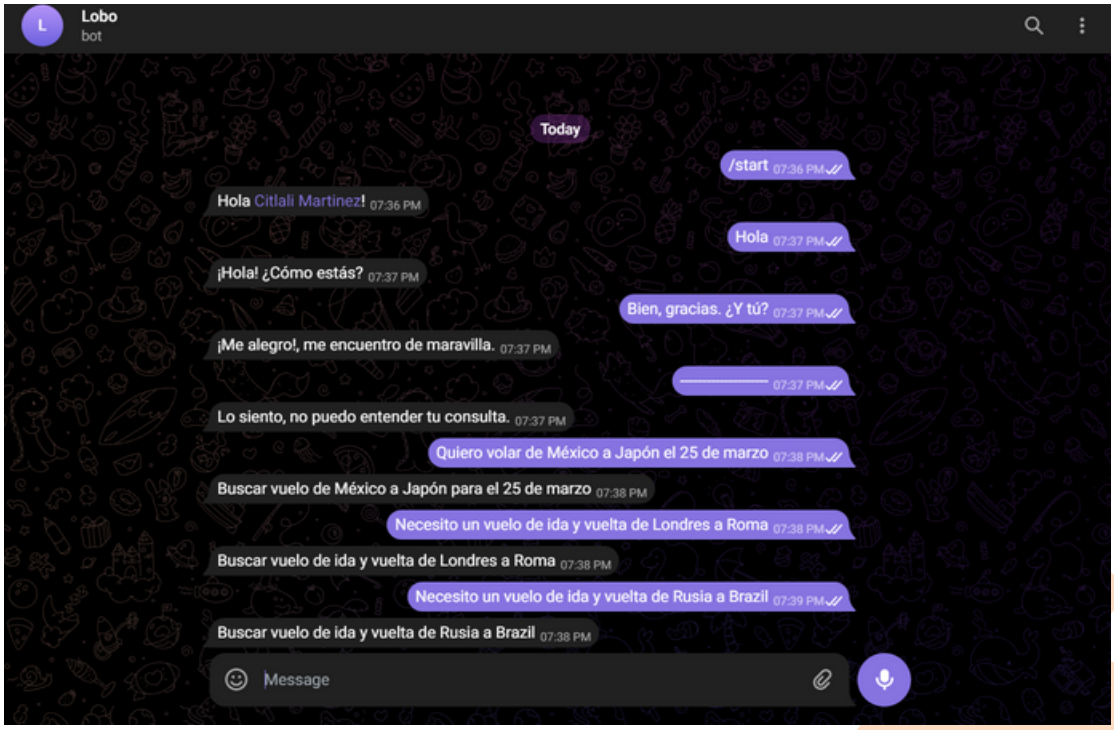
```
if re.search(patron_origen_destino_fecha, message_text):
origen_destino_fecha = re.search(patron_origen_destino_fecha, message_text)
origen = origen_destino_fecha.group(1)
destino = origen_destino_fecha.group(2)
fecha = origen_destino_fecha.group(3)
await update.message.reply_text(f"Buscar vuelo de {origen} a {destino} para el {fecha}")
elif re.search(patron_precio, message_text):
precio = re.search(patron_precio, message_text)
origen = precio.group(1)
destino = precio.group(2)
await update.message.reply_text(f"Consultar precio de vuelo de {origen} a {destino}")

elif re.search(patron_ida_vuelta, message_text):
ida_vuelta = re.search(patron_ida_vuelta, message_text)
origen = ida_vuelta.group(1)
destino = ida_vuelta.group(2)
await update.message.reply_text(f"Buscar vuelo de ida y vuelta de {origen} a {destino}")
else:
await update.message.reply_text("Lo siento, no puedo entender tu consulta.")


def main() -> None:
"""Iniciar el bot."""
application = Application.builder().token("6520442896:AAEAEcqTLXdLNCWnvdH0M4dBH8wNp-bt6jM").build()

application.add_handler(CommandHandler("start", start))
application.add_handler(CommandHandler("help", help_command))
application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, echo))

application.run_polling(allowed_updates=Update.ALL_TYPES)


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

PRÁCTICA

BOT EN FUNCIONAMIENTO



CONSULTAS REALIZADAS EN TELEGRAM



CONCLUSIONES

Al procesar las consultas que se ingresan en el chat del bot de Telegram, es crucial tener cuidado con la lógica de procesamiento para evitar respuestas duplicadas o no deseadas. Al realizar bien las expresiones regulares en el código se logra evitar las respuestas duplicadas y mejorar la experiencia del usuario. Es importante siempre revisar el código para garantizar un funcionamiento óptimo del bot y proporcionar expresiones correctas.