```
'''
TELEGRAM CHATBOT WITH MACHINE LEARNING
Implementation of a Chatbot to work with Pizza orders.
Built with Telegram API and use of 2 approches with Machine Learning models:

* To predict and suggest pizza toppings to clients;
* To analyze post-sale feedback and comments.


Models implemented:

* Random Forest;
* Logistic regression;
* Naive Bayes;
* KNN;
* Term Frequency-Inverse Document Frequency (TF-IDF);
* Bag of words (BOW).
The BOW and TF-IDF was made with a tweet file.


In this file you can analyze the implementation of the Bot and use of the models built in the Jupyter file.


Important: This project was built for a Brazilian client, so the features, objects, and models have
their names in Portuguese.
'''


# Libraries
import logging
import math
import psycopg2
import pickle
import pandas as pd
import re
import datetime
from telegram import ReplyKeyboardMarkup, ChatAction
from telegram.ext import (Updater, CommandHandler, MessageHandler, Filters, ConversationHandler, BaseFilter)
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer
from unicodedata import normalize
import requests

# Standard data for chatbot
token = 'telegram_tolken'
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', level=logging.INFO)
logger = logging.getLogger(__name__)

# Postgres Database
database = 'postgres'
username = 'postgres'
password = '123456'

# Predictors and word processing
caminho = 'path'

preditor_pizza = pickle.load(open(caminho + '/preditor_pizza.pkl', 'rb'))
preditor_avaliacao = pickle.load(open(caminho + '/preditor_avaliacao_bow.pkl', 'rb'))
bow = pickle.load(open(caminho + '/bow.pkl', 'rb'))
stop_words = list(pd.read_csv(caminho + '/stop_word.csv', encoding='Latin1', sep =';'))


# List of interactions for use in MAIN
PRIM_INTERACAO, PRIM_ITERACAO_DIGITE, PRIM_ITERACAO_DIGITE2, PRIM_ITERACAO_DIGITE3, \
CARDAPIO_OPCOES, CHOOSING_GENDER, PIZZA_OPCOES, DOCES_OPCOES, ADICIONAR, FINALIZAR_COMPRA, FORMAS_PGTO, \
MOTIVO_FIM, FIM_PEDIDO, BEBIDAS_OPCOES, TESTE, CAD_NOME, CAD_CPF, CAD_CEP, CAD_GENERO, CAD_IDADE, \
CAD_TEL, FIM_CAD, TROCO_PARA, TAMANHO, NUM_SABORES, ADD_ITEM, QTD_PEDIDO, VALID_RUA, DIGITA_NUMERO, \
DIGITA_COMPLEMENTO, TROCA_RUA, CONFIRMACAO, ALTERAR_CAD, OBSERVACAO,NOTA_PIZZA = range(35)


# Button List
reply_keyboard = [['QUANTAS 🍕 ❓', 'CARDÁPIO 📖'],['INFORMAÇÕES 💡', 'FINALIZAR 🙋']]
#reply_keyboard = [['CALCULADORA 🧮', 'CARDÁPIO 📖'],['CADASTRO 📝','INFORMAÇÕES 💡', 'FINALIZAR 🙋']]
markup = ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True,resize_keyboard=True)

cardapio_keyboard = [['TRADICIONAIS 🍕', 'DOCES 🍫'],['BEBIDAS 🥤']]
markup_card = ReplyKeyboardMarkup(cardapio_keyboard, one_time_keyboard=True,resize_keyboard=True)

tamanho_keyboard = [['GRANDE - 8 PEDAÇOS'],['BROTO - 4 PEDAÇOS']]
tamanho_markup = ReplyKeyboardMarkup(tamanho_keyboard, one_time_keyboard=True,resize_keyboard=True)

sabores_keyboard = [['1 SABOR', '2 SABORES']]
```

```
sabores_markup = ReplyKeyboardMarkup(sabores_keyboard, one_time_keyboard=True,resize_keyboard=True)

adic_rem_keyboard = [['ADICIONAR ☝ ', 'REMOVER ❌'],['FINALIZAR PEDIDO ✅']]
markup_adic = ReplyKeyboardMarkup(adic_rem_keyboard, one_time_keyboard=True,resize_keyboard=True)

confir_pedido = [['ADICIONAR ☝ ', 'REMOVER ❌'],['CONFIRMAR COMPRA ✅']]
markup_confir = ReplyKeyboardMarkup(confir_pedido, one_time_keyboard=True,resize_keyboard=True)

pgto_keyboard = [['CRÉDITO', 'DÉBITO','DINHEIRO']]
markup_pgto = ReplyKeyboardMarkup(pgto_keyboard, one_time_keyboard=True,resize_keyboard=True)

rua_keyboard = [['SIM','NÃO']]
markup_rua = ReplyKeyboardMarkup(rua_keyboard, one_time_keyboard=True,resize_keyboard=True)

confirm_keyboard = [['SIM!','NÃO!']]
markup_confirm = ReplyKeyboardMarkup(confirm_keyboard, one_time_keyboard=True,resize_keyboard=True)

cadastro_keyboard = [['NOME','CPF','NASCIMENTO'],['CEP','RUA','NUMERO'],['COMPLEMENTO','TELEFONE','GENERO']]
markup_cadastro = ReplyKeyboardMarkup(cadastro_keyboard, one_time_keyboard=True,resize_keyboard=True)

# Database
def criando_bdados(database, username, password):

    # The first step is to download PGAdmin and create the chatizza database. After that the code creates the data.
    con = psycopg2.connect(host='localhost', database=database, user=username, password=password)
    cur = con.cursor()

    # creating the registration table
    sql = 'CREATE TABLE IF NOT EXISTS cadastro (TELEGRAMID int primary key,' \
          'NOME varchar (30),' \
          'CPF varchar (20),' \
          'DTNASC varchar(10),' \
          'GENERO varchar (15),' \
          'CEP varchar(9),' \
          'RUA varchar(200),' \
          'NUMERO varchar(10),' \
          'COMPLEMENTO varchar(50),' \
          'TELEFONE varchar (15) NOT NULL)'
    cur.execute(sql)

    # creating the pizza table
    sql = 'CREATE TABLE IF NOT EXISTS pizzas (ID_PRODUTO int primary key,' \
          'NOME varchar(30),' \
          'TIPO varchar(15),' \
          'VALOR_REAL money)'
    cur.execute(sql)

    # creating the ingredients table
    #sql = 'CREATE TABLE IF NOT EXISTS ingredientes (ID_PRODUTO int primary key,' \
    #      'TELEGRAMID int, ' \
    #      'NOME varchar (50),' \
    #      'QUANTIDADE int,' \
    #      'VALOR_REAL money)'
    #cur.execute(sql)

    # creating the sweet pizza table
    sql = 'CREATE TABLE IF NOT EXISTS doces (ID_PRODUTO int primary key,' \
          'NOME varchar(30),' \
          'TIPO varchar(15),' \
          'VALOR_REAL money)'
    cur.execute(sql)

    # creating the drinks table
    sql = 'CREATE TABLE IF NOT EXISTS bebidas (ID_PRODUTO int primary key,' \
          'NOME varchar(30),' \
          'TIPO varchar(15),' \
          'VALOR_REAL money)'
    cur.execute(sql)

    # creating the cart items table
    sql = 'CREATE TABLE IF NOT EXISTS carrinho (item_pedido serial,' \
          'TELEGRAMID int,' \
          'ID_PRODUTO int,' \
          'TIPO varchar(30),' \
          'TIPO2 varchar(30),' \
          'TAMANHO varchar(20),' \
          'NOME varchar (50),' \
          'QUANTIDADE float,' \
          'VALOR_REAL money)'
    cur.execute(sql)

    # creating the orders table
    sql = 'CREATE TABLE IF NOT EXISTS pedidos (ID_PEDIDO int,' \
          'ID_PEDIDO_CLIENTE int,' \
          'DATA date,' \
          'HORA time,' \
```

```
            'TELEGRAMID int,' \
            'ID_PRODUTO int,' \
            'TIPO varchar(30),' \
            'TIPO2 varchar(30),' \
            'TAMANHO varchar(20),' \
            'NOME varchar (100),' \
            'QUANTIDADE float,' \
            'FORMA_PAGTO varchar(30),' \
            'OBSERVACAO varchar(500),' \
            'VALOR_UNITARIO money,' \
            'VALOR_TOTAL money,' \
            'VALOR_COMPRA money,' \
            'TROCO money)'
    cur.execute(sql)

    # creating the table of evaluations and feedbacks
    sql = 'CREATE TABLE IF NOT EXISTS avaliacoes (ID_PEDIDO int,' \
          'ID_PEDIDO_CLIENTE int,' \
          'TELEGRAMID int,' \
          'NOTA_ATENDIMENTO int,' \
          'AVALIACAO_ATENDIMENTO VARCHAR(500),' \
          'NLP_ATENDIMENTO VARCHAR(20),' \
          'NOTA_PIZZA int,' \
          'AVALIACAO_PIZZA VARCHAR(500),' \
          'NLP_PIZZA VARCHAR(20))'
    cur.execute(sql)

    con.commit()
    con.close()

# Functions (def)
# DEFs from database
def alteracoes_bdados(database, codsql, username, password):
    con = psycopg2.connect(host='localhost', database=database, user=username, password=password)
    cur = con.cursor()
    cur.execute(codsql)
    con.commit()
    con.close()


def select_bdados(database, codsql, username, password):
    con = psycopg2.connect(host='localhost', database=database, user=username, password=password)
    cur = con.cursor()
    cur.execute(codsql)
    recset = cur.fetchall()
    return recset


def update_bdados(database, codsql, username, password):
    con = psycopg2.connect(host='localhost', database=database, user=username, password=password)
    cur = con.cursor()
    cur.execute(codsql)
    con.commit()
    con.close()
    #recset = cur.fetchall()
    #return recset

# DEFs for start and error
class StartFilter(BaseFilter):
    def filter(self, message):
        if message.text.lower() in ['oi','ola','pizza','quero']:
            return True
        else:
            return False

class EndFilter(BaseFilter):
    def filter(self, message):
        if message.text.lower() in ['finalizar']:
            return True
        else:
            return False

def start(update, context):
    user_data = context.user_data
    user_data.clear()
    logger.info('alguém iniciou uma conversa')
    update.message.reply_text('Olá! Só um momento que estou buscando seu cadastro.'"\n"
                              "Caso queira finalizar o atendimento basta digitar 'finalizar' a qualquer momento.")
    # SQL
    user_data['ID_TELEGRAM'] = str(update.message.from_user['id'])
    user_data['NICKNAME'] = str(update.message.from_user['first_name'])
    select = 'select * from cadastro where telegramID = ' + user_data['ID_TELEGRAM']
    resultado = select_bdados(database, select, username, password)
    if len(resultado) == 0:
        update.message.reply_text(
            'Olá '+ user_data['NICKNAME'] +', Não localizamos seu cadastro. Precisaremos de algumas informações suas, ok?')
```

```
        return cadastro(update)
    return saudacao(update, context)


def error(update, context):
    """Log Errors caused by Updates."""
    logger.warning('Update "%s" caused error "%s"', update, context.error)

# DEFs for register
def cadastro(update):
    update.message.reply_text('É rapidinho.''\n''Por favor, me informe seu nome completo.')
    return CAD_NOME


def armazena_nome(update, context):
    user_data = context.user_data
    user_data['NOME'] = update.message.text.upper()
    update.message.reply_text("\n" "NOME - {}" "\n"
                              "E qual seu CPF {}? Por favor, preecha somente números".format(user_data['NOME'], \
                                                                                             user_data['NICKNAME']))
    return CAD_CPF


def armazena_cpf(update, context):
    user_data = context.user_data
    try:
        int(update.message.text)
        if 11 <= len(update.message.text) <= 14:
            user_data['CPF'] = update.message.text
            update.message.reply_text("Certo." "\n" "CPF - {}" "\n" "\n"
                                      "Já está terminando, prometo." "\n" "\n"
                                      "E qual seu CEP? Por favor, digite apenas números".format(user_data['CPF']))
            return CAD_CEP
        else:
            update.message.reply_text(
                'Não entendi, quantidade de caracteres incorreta, cpf possui 11 digitos e cnpj 14. Qual seu CPF?')
            return CAD_CPF
    except:
        update.message.reply_text('Não entendi, digite apenas números (por exemplo 39423456799). Qual seu CPF?')
        return CAD_CPF


def armazena_cep(update, context):
    user_data = context.user_data
    user_data['CEP'] = update.message.text
    user_data['VALID_END'] = 0
    try:
        int(update.message.text)
        consulta_end = requests.get('https://viacep.com.br/ws/' + user_data['CEP'] + '/json/').json()
        user_data['RUA'] = str(consulta_end['logradouro'].replace('\'', ''))
        user_data['BAIRRO'] = str(consulta_end['bairro'].replace('\'', ''))
        user_data['CIDADE'] = str(consulta_end['localidade'].replace('\'', ''))
        user_data['UF'] = str(consulta_end['uf'].replace('\'', ''))
        if len(update.message.text) == 8:
            update.message.reply_text("\n" "CEP - {}" "\n"
                                      "Sua rua é {} ?".format(user_data['CEP'],user_data['RUA']),
                                      reply_markup=markup_rua)
            return VALID_RUA

        else:
            update.message.reply_text(
            'Não entendi, quantidade de caracteres incorreta (tem que ser 8). Qual seu CEP?')
            return CAD_CEP
    except:
        update.message.reply_text('Não entendi, digite apenas números sem pontos ou traço (por exemplo 17012543)." "\n"'
                                  '"Qual seu CEP?')
        return CAD_CEP


def pass_numero(update, context):
    update.message.reply_text('Qual o número da residência? Por favor digite apenas números.')
    return DIGITA_NUMERO

def troca_rua(update, context):
    user_data = context.user_data
    update.message.reply_text('Beleza, qual o nome correto da rua?')
    return TROCA_RUA

def guarda_rua(update, context):
    user_data = context.user_data
    user_data['RUA'] = update.message.text
    update.message.reply_text('Certo, ajustamos o nome da rua aqui." "\n" '
                              '"Qual o número da residência? Por favor digite apenas números.')
    return DIGITA_NUMERO

def armazena_numero(update, context):
```

```
    user_data = context.user_data
    try:
        int(update.message.text)
        user_data['NUMERO'] = update.message.text
        update.message.reply_text("\n" "NUMERO - {}" "\n" "Certo, tem algum complemento?" "\n"
                                  "Se não tiver, só digitar 'não'.".format(user_data['NUMERO']))
        return DIGITA_COMPLEMENTO
    except:
        update.message.reply_text('Não entendi, digite apenas números. Qual o número da residência?')
        return DIGITA_NUMERO

def armazena_complemento(update, context):
    user_data = context.user_data
    user_data['COMPLEMENTO'] = update.message.text
    if user_data['VALID_END'] == 1:
        return valid_cadastro(update, context)
    else:
        update.message.reply_text("\n" "COMPLEMENTO - {}" "\n"
                                  "Ótimo.E qual seu gênero?" "\n"
                                  "Por favor, responda 'HOMEM', 'MULHER' ou 'OUTRO'.".format(user_data['COMPLEMENTO']))
        return CAD_GENERO

def armazena_genero(update, context):
    user_data = context.user_data
    lista = ['HOMEM','MULHER','OUTRO']
    if update.message.text.upper() in lista:
        user_data['GENERO'] = update.message.text.upper()
        update.message.reply_text("\n" "GENERO - {}" "\n" "Só mais duas informações" "\n"
                                  "Qual sua data de nascimento?" "\n"
                                  "Digite no formato DD/MM/AAAA (dia,mes,ano).".format(user_data['GENERO']))
        return CAD_IDADE
    else:
        update.message.reply_text("Não entendi, por favor, responda 'HOMEM', 'MULHER' ou 'OUTRO'")
        return CAD_GENERO


def armazena_idade(update, context):
    user_data = context.user_data
    try:
        user_data['DTNASC'] = datetime.datetime.strptime(update.message.text, '%d/%m/%Y')
        idade = int(((datetime.datetime.today()-user_data['DTNASC']).days/365.25))
        update.message.reply_text("\n" "Você tem {} anos."'\n' "E por último, qual é seu telefone com DDD?" '\n'
                                  "Digite apenas números sem espaço".format(idade))
        return CAD_TEL
    except:
        update.message.reply_text('Não entendi, digite no formato DD/MM/AAAA (dia,mes,ano)')
        return CAD_IDADE


def armazena_tel(update, context):
    user_data = context.user_data
    try:
        print(update.message.text)
        user_data['TELEFONE'] = update.message.text
        user_data['NICKNAME'] = str(update.message.from_user['first_name'])
        user_data['ID_TELEGRAM'] = str(update.message.from_user['id'])
        user_data['DTNASC'] = str(user_data['DTNASC'])[0:10]
        select = "insert into cadastro (telegramid, nome, cpf, dtnasc, genero, cep, rua, " \
                 "numero, complemento, telefone) values ("+str(user_data['ID_TELEGRAM'])+",'"+\
                 str(user_data['NOME'])+"','"+str(user_data['CPF'])+"','"+str(user_data['DTNASC'])+\
                 "','"+str(user_data['GENERO'])+"','"+str(user_data['CEP'])+"','"+str(user_data['RUA'])+\
                 "','"+str(user_data['NUMERO'])+"','"+str(user_data['COMPLEMENTO'])+"','"+\
                 str(user_data['TELEFONE'])+"')"

        print(select)
        alteracoes_bdados(database, select, username, password)
        update.message.reply_text("\n" "TELEFONE - {}" "\n"
                                  "Certo, muito obrigado pelas informações." "\n"
                                  "Vamos a parte boa agora 😊".format(user_data['TELEFONE']))
        return saudacao(update, context)
    except:
        update.message.reply_text('Não entendi, digite apenas números. Qual seu telefone?')
        return CAD_TEL


def saudacao(update, context):
    user_data = context.user_data

    deleta_historico = "delete from carrinho where TELEGRAMID ="+ user_data['ID_TELEGRAM']
    alteracoes_bdados(database, deleta_historico, username, password)

    select = "select dtnasc, genero, cep, rua, numero, complemento from cadastro where TELEGRAMID ="+\
             user_data['ID_TELEGRAM']

    info_cadastro = list(str(select_bdados(database, select, username, password))[2:-2].replace('\'', '').split(', '))
    user_data['DTNASC'], user_data['GENERO'], user_data['CEP'], user_data['RUA'], user_data['NUMERO'],\
```

```
    user_data['COMPLEMENTO'] = info_cadastro[0:6]

    user_data['IDADE'] = int(((datetime.datetime.today() -
                               datetime.datetime.strptime(user_data['DTNASC'],'%Y-%m-%d')).days / 365.25))

    user_data['HORA'] = datetime.datetime.now().hour

    user_data['DATA_ATENDIMENTO'] = datetime.datetime.now().strftime("%Y-%m-%d")

    user_data['HORA_ATENDIMENTO'] = datetime.datetime.now().strftime("%H:%M:%S")

    if datetime.datetime.now().minute <=30:
        user_data['MIN_ATE_30'] = 1
        user_data['MIN_MAIS_30'] = 0
    else:
        user_data['MIN_ATE_30'] = 0
        user_data['MIN_MAIS_30'] = 1

    if user_data['GENERO'] == 'HOMEM':
        user_data['HOMEM'] = 1
        user_data['MULHER'] = 0
        user_data['OUTRO'] = 0
    elif user_data['GENERO'] == 'MULHER':
        user_data['HOMEM'] = 0
        user_data['MULHER'] = 1
        user_data['OUTRO'] = 0
    else:
        user_data['HOMEM'] = 0
        user_data['MULHER'] = 0
        user_data['OUTRO'] = 1

    if int(user_data['IDADE']) > 70:
        user_data['MAIS_70'] = 1
        user_data['ATE_23'] = 0
        user_data['ATE_30'] = 0
        user_data['ATE_40'] = 0
        user_data['ATE_50'] = 0
        user_data['ATE_70'] = 0
    elif int(user_data['IDADE']) < 24:
        user_data['MAIS_70'] = 0
        user_data['ATE_23'] = 1
        user_data['ATE_30'] = 0
        user_data['ATE_40'] = 0
        user_data['ATE_50'] = 0
        user_data['ATE_70'] = 0
    elif int(user_data['IDADE']) < 31:
        user_data['MAIS_70'] = 0
        user_data['ATE_23'] = 0
        user_data['ATE_30'] = 1
        user_data['ATE_40'] = 0
        user_data['ATE_50'] = 0
        user_data['ATE_70'] = 0
    elif int(user_data['IDADE']) < 41:
        user_data['MAIS_70'] = 0
        user_data['ATE_23'] = 0
        user_data['ATE_30'] = 0
        user_data['ATE_40'] = 1
        user_data['ATE_50'] = 0
        user_data['ATE_70'] = 0
    elif int(user_data['IDADE']) < 51:
        user_data['MAIS_70'] = 0
        user_data['ATE_23'] = 0
        user_data['ATE_30'] = 0
        user_data['ATE_40'] = 0
        user_data['ATE_50'] = 1
        user_data['ATE_70'] = 0
    else:
        user_data['MAIS_70'] = 0
        user_data['ATE_23'] = 0
        user_data['ATE_30'] = 0
        user_data['ATE_40'] = 0
        user_data['ATE_50'] = 0
        user_data['ATE_70'] = 1

    update.message.reply_text(
        "Olá " + user_data['NICKNAME'] + ", nós somos da Chatizza! Eu vou ajudar você a fazer seu pedido." '\n'
        "Habilite o campo de botões, clicando ao lado do emoji no campo de mensagem ok?"'\n'
        "Por onde começamos?",
        reply_markup=markup)
    return PRIM_INTERACAO

def altera_cadastro(update, context):
    #user_data = context.user_data
    update.message.reply_text('O seu cadastro atual é:')
    update.message.reply_text('O que deseja alterar?', reply_markup=markup_cadastro)
    return ALTERAR_CAD
```

```
def guarda_alteracao(update, context):
    user_data = context.user_data
    print(user_data)
    #select = "insert into cadastro (telegramid, nome, cpf, dtnasc, genero, cep, rua, numero, " \
    #         "complemento, telefone) values (" + str(
    #    user_data['ID_TELEGRAM']) + ",'" + str(user_data['NOME']) + "','" + str(user_data['CPF']) + "','" + str(
    #   user_data['DTNASC']) + "','" + str(user_data['GENERO']) + "','" + str(user_data['CEP']) + "','" + str(
    # user_data['RUA']) + "','" + str(user_data['NUMERO']) + "','" + str(user_data['COMPLEMENTO']) + "','" + str(
    # user_data['TELEFONE']) + "')"

# DEF for informations
def infos_choice(update, context):
    update.message.reply_text('- Nosso horário de atendimento é das 18:00 às 00:00. ''\n'
                              '- Nossas entregas funcionam até as 22:00.''\n'
                              '\n'
                              'Podemos ajudá-lo em algo mais?''\n'
                              'Caso não precise de mais nada, basta clicar em "FINALIZAR"',
                              reply_markup=markup)
    return PRIM_INTERACAO


# DEFs for calculator
def dimensionamento_choice(update, context):
    update.message.reply_text('Certo, vamos te ajudar a calcular quantas pizzas você precisará! 😃''\n'
                              'São quantos homens?')
    return PRIM_ITERACAO_DIGITE


def dimensionamento_homem(update, context):
    user_data = context.user_data
    try:
        int(update.message.text)
        user_data['HOMEM'] = update.message.text
        update.message.reply_text("\n" "{} Homens" "\n" "Certo, e quantas mulheres? ".format(user_data['HOMEM']))
        return PRIM_ITERACAO_DIGITE2
    except:
        update.message.reply_text('Não entendi, digite apenas números (por exemplo 6). Quantos homens irão?')
        return PRIM_ITERACAO_DIGITE


def dimensionamento_mulher(update, context):
    user_data = context.user_data
    try:
        int(update.message.text)
        user_data['MULHER'] = update.message.text
        update.message.reply_text("\n" "{} Mulheres" "\n" "Okay, e quantas crianças? ".format(user_data['MULHER']))
        return PRIM_ITERACAO_DIGITE3
    except:
        update.message.reply_text('Não entendi, digite apenas números (por exemplo 4). Quantas mulheres?')
        return PRIM_ITERACAO_DIGITE2


def dimensionamento_crianca(update, context):
    user_data = context.user_data
    try:
        int(update.message.text)
        user_data['CRIANCA'] = update.message.text
        update.message.reply_text(
            "Então pelo que me disse são: " "\n"
            "HOMEM: {}" "\n"
            "MULHER: {}" "\n"
            "CRIANÇA: {}".format(user_data['HOMEM'],user_data['MULHER'],user_data['CRIANCA']))

        update.message.reply_text('Olha, pelas minhas contas, você precisará de no mínimo {} pizzas'.format(math.ceil(
            (int(user_data['HOMEM']) * 4 + int(user_data['MULHER']) * 3 + int(user_data['CRIANCA']) * 2) / 8)),
            reply_markup=markup)
        return PRIM_INTERACAO
    except:
        update.message.reply_text('Não entendi, digite apenas números (por exemplo 3). E quantas crianças?')
        return PRIM_ITERACAO_DIGITE3

# DEFs for menu
def cardapio_choice(update, context):
    update.message.reply_text('Muito bom, O que você deseja ver?', reply_markup=markup_card)
    return CARDAPIO_OPCOES


def tamanhos(update, context):
    update.message.reply_text('Legal, você quer uma pizza de qual tamanho?', reply_markup=tamanho_markup)
    return TAMANHO


def sabores(update, context):
    user_data = context.user_data
    if update.message.text == 'GRANDE - 8 PEDAÇOS':
```

```
        user_data['TAMANHO'] = 'GRANDE'
    else:
        user_data['TAMANHO'] = 'BROTO'
    update.message.reply_text('Quantos sabores?', reply_markup=sabores_markup)
    return NUM_SABORES


def opcoes_pizzas(update, context):
    user_data = context.user_data
    user_data['CARDAPIO'] = 'PIZZAS'
    select = "select id_produto, nome, valor_real from "+user_data['CARDAPIO']
    cardapio = select_bdados(database, select, username, password)
    update.message.reply_text(
        'Escolha uma de nossas opções abaixo e digite o código da pizza desejada.' '\n'
        'Posso te dar uma sugestão de pizza?' '\n' +
        'Que tal uma de {} ?'.format(str(preditor_pizza.predict([[user_data['CEP'], user_data['HORA'],
                                                                  user_data['HOMEM'], user_data['MULHER'],
                                                                  user_data['OUTRO'], user_data['MAIS_70'],
                                                                  user_data['ATE_23'], user_data['ATE_30'],
                                                                  user_data['ATE_40'], user_data['ATE_50'],
                                                                  user_data['ATE_70'], user_data['MIN_MAIS_30'],
                                                                  user_data['MIN_ATE_30']]])[0])) + '\n'
        'CÓDIGO - SABOR - PREÇO: ' '\n')

    for row in cardapio:
        update.message.reply_text(str(row)[1:-1].upper().replace(', ',' - ').replace('\'', ''))
    if update.message.text == '2 SABORES':
        user_data['TIPO2'] = 'MEIA'
        user_data['PARTE'] = '1'
    else:
        user_data['TIPO2'] = 'INTEIRA'
        user_data['PARTE'] = '2'
        return QTD_PEDIDO

    return ADD_ITEM

def quantidades(update, context):
    user_data = context.user_data
    user_data['PEDIDO'] = update.message.text
    if user_data['CARDAPIO'] == 'PIZZAS':
        update.message.reply_text("Certo, quantas deste sabor? Digite apenas números")
    if user_data['CARDAPIO'] == 'DOCES':
        update.message.reply_text("Certo, quantas deste sabor? Digite apenas números")
    if user_data['CARDAPIO'] == 'BEBIDAS':
        update.message.reply_text("Certo, quantas unidades? Digite apenas números")
    return ADD_ITEM


def add_item(update, context):
    user_data = context.user_data
    try:
        int(update.message.text)
        if user_data['CARDAPIO'] == 'DOCES' or user_data['CARDAPIO'] == 'BEBIDAS' :
            insert = "insert into carrinho(TELEGRAMID, id_produto, quantidade, tipo, tipo2, " \
                     "tamanho, nome, valor_real) select " + str(user_data['ID_TELEGRAM']) + ", id_produto, "\
                     + update.message.text +", tipo, '" \
                     + str(user_data['TIPO2']) + "','" + str(user_data['TAMANHO']) + "', nome, valor_real from " \
                     + user_data['CARDAPIO'] + " where id_produto =" + str(user_data['PEDIDO'])

            alteracoes_bdados(database, insert, username, password)
            update.message.reply_text("Certo, pedido adicionado. Deseja mais alguma coisa?", reply_markup=markup_adic)
            return ADICIONAR

        elif user_data['CARDAPIO'] == 'PIZZAS':
            if user_data['TIPO2'] == 'MEIA' and user_data['PARTE'] == '1':
                insert = "insert into carrinho(TELEGRAMID, id_produto, quantidade, tipo, tipo2, tamanho, nome, " \
                         "valor_real) select " + str(user_data['ID_TELEGRAM']) + ", id_produto, 0.5, tipo, '" \
                         + str(user_data['TIPO2']) + "','" + str(user_data['TAMANHO']) + \
                         "', nome, valor_real from " + user_data['CARDAPIO'] + " where id_produto =" \
                         + update.message.text

                alteracoes_bdados(database, insert, username, password)
                user_data['PARTE'] = '2'
                select = "select nome from "+user_data['CARDAPIO']+" where id_produto =" + update.message.text
                pedido = select_bdados(database, select, username, password)
                select = "select id_produto, nome, valor_real from "+user_data['CARDAPIO']
                cardapio = select_bdados(database, select, username, password)
                update.message.reply_text('Certo, meia pizza de {}, Qual o sabor da outra metade?' '\n'
                                          'CÓDIGO, SABOR, PREÇO: ' '\n'.format(str(pedido)[3:-4].upper()))
                for row in cardapio:
                    update.message.reply_text(str(row)[1:-1].upper().replace(', ', ' - ').replace('\'', ''))
                return ADD_ITEM

            else:
                if user_data['TIPO2'] == 'MEIA':
                    insert = "insert into carrinho(TELEGRAMID, id_produto, quantidade, tipo, tipo2, " \
```

```
                            "tamanho, nome, valor_real) select " + str(user_data['ID_TELEGRAM']) \
                            + ", id_produto, 0.5, tipo, '" + str(user_data['TIPO2']) + "','" \
                            + str(user_data['TAMANHO']) + "', nome, valor_real from " \
                            + user_data['CARDAPIO'] + " where id_produto =" + update.message.text

                        alteracoes_bdados(database, insert, username, password)
                    else:
                        insert = "insert into carrinho(TELEGRAMID, id_produto, quantidade, tipo, tipo2, tamanho, " \
                                 "nome, valor_real) select " + str(user_data['ID_TELEGRAM']) + ", id_produto, "\
                                 + update.message.text+", tipo, '" + str(user_data['TIPO2']) + "','" \
                                 + str(user_data['TAMANHO']) + "', nome, valor_real from " \
                                 + user_data['CARDAPIO'] + " where id_produto =" + str(user_data['PEDIDO'])

                        alteracoes_bdados(database, insert, username, password)

                    update.message.reply_text("Prontinho, anotado", reply_markup=markup_adic)
                    return ADICIONAR

            else:
                 insert = "DELETE FROM carrinho WHERE item_pedido IN " \
                 "(SELECT item_pedido FROM ( SELECT item_pedido, RANK() OVER(PARTITION BY telegramid " \
                          "ORDER BY item_pedido) AS pedido "\
                 "FROM carrinho ) AS RKG WHERE RKG.pedido = " + update.message.text + " AND telegramid = " \
                           + str(user_data['ID_TELEGRAM']) + ") AND telegramid = " + str(user_data['ID_TELEGRAM'])

                 alteracoes_bdados(database, insert, username, password)

                 update.message.reply_text("Pronto, item removido dos pedidos. Deseja mais alguma coisa?",
                                           reply_markup=markup_adic)
                 return ADICIONAR

        except:
            update.message.reply_text('Não entendi, digite apenas números (por exemplo: 1)')
            return ADD_ITEM


    def remove_item(update, context):
        user_data = context.user_data
        user_data['CARDAPIO'] = 'REMOVER'

        select = "SELECT RANK() OVER( PARTITION BY telegramid ORDER BY item_pedido ) as codigo , " \
                 "nome, quantidade, valor_real*quantidade as valor FROM carrinho where telegramid = "\
                 + str(user_data['ID_TELEGRAM'])

        cardapio = select_bdados(database, select, username, password)

        update.message.reply_text(
            'Certo, qual dos itens abaixo você gostaria de remover? ' '\n'
            'Digite apenas o código.' '\n' 'CÓDIGO - NOME - QUANTIA - PREÇO: ' '\n')

        for row in cardapio:
            update.message.reply_text(str(row)[1:-1].upper().replace(', ', ' - ').replace('\'', ''))
        return ADD_ITEM


    def opcoes_doces(update, context):
        user_data = context.user_data
        user_data['CARDAPIO'] = 'DOCES'
        user_data['TAMANHO'] = 'UNICO'
        user_data['TIPO2'] = 'UNICO'
        select = "select id_produto, nome, valor_real from "+ user_data['CARDAPIO']
        cardapio = select_bdados(database, select, username, password)

        update.message.reply_text(
            'Escolha uma de nossas opções abaixo e digite o código da pizza desejada.' '\n'
            'CÓDIGO - SABOR - PREÇO: ' '\n')

        for row in cardapio:
            update.message.reply_text(str(row)[1:-1].upper().replace(', ', ' - ').replace('\'', ''))
        return QTD_PEDIDO


    def opcoes_bebidas(update, context):
        user_data = context.user_data
        user_data['CARDAPIO'] = 'BEBIDAS'
        user_data['TAMANHO'] = 'UNICO'
        user_data['TIPO2'] = 'UNICO'
        select = "select id_produto, nome, valor_real from " + user_data['CARDAPIO']
        cardapio = select_bdados(database, select, username, password)

        update.message.reply_text('Nada melhor que uma bebida pra acompanhar a pizza né? ' '\n'
                                  '😎, Qual bebida você quer colocar no pedido? Pode só me dizer o número do item:.' '\n'
                                  'CÓDIGO - BEBIDA - PREÇO: ' '\n')

        for row in cardapio:
            update.message.reply_text(str(row)[1:-1].upper().replace(', ', ' - ').replace('\'', ''))
```

```
    return QTD_PEDIDO


def fim_pedido(update, context):
    user_data = context.user_data
    select = "select  quantidade, nome, valor_real, valor_real*quantidade as total from " \
             "carrinho where telegramid = " + str(user_data['ID_TELEGRAM'])

    cardapio = select_bdados(database, select, username, password)

    select = "select sum(valor_real*quantidade) as valortotal from carrinho where telegramid = " \
             + str(user_data['ID_TELEGRAM'])

    user_data['VALOR_TOTAL'] = str(select_bdados(database, select, username, password))[3:-4]#.replace('\'', '')

    update.message.reply_text('Abaixo segue lista de pedidos' '\n' 'QUANTIA - NOME - VALOR UNIDADE - SUB TOTAL: ' '\n')

    for row in cardapio:
        update.message.reply_text(str(row)[1:-1].upper().replace(', ', ' - ').replace('\'', ''))
    update.message.reply_text('Seus pedidos estão totalizando {}. Confirmar a compra?'.format(user_data['VALOR_TOTAL']),
                              reply_markup=markup_confir)
    return FINALIZAR_COMPRA

def observacao(update, context):
    update.message.reply_text("Gostaria de fazer alguma observacao para o pedido?" "\n"
                              "Retirar algum ingrediente ou mudar o corte da pizza, estou anotando 😉" )
    return OBSERVACAO

def salva_obs(update, context):
    user_data = context.user_data
    user_data['OBSERVACAO'] = update.message.text
    update.message.reply_text("Beleza... Seu pedido já está me deixando com fome também 😅" "\n")
    return valid_cadastro(update, context)

def valid_cadastro(update,context):
    user_data = context.user_data
    user_data['VALID_END'] = 1
    update.message.reply_text('Vamos fazer uma confirmação de endereço, tudo bem?'"\n"
                              'O endereço para entrega está correto?'"\n"
                              'RUA: {}'"\n"
                              'NUMERO: {}'"\n"
                              'COMPLEMENTO: {}'.format(user_data['RUA'],
                                                       user_data['NUMERO'],
                                                       user_data['COMPLEMENTO']), reply_markup=markup_confirm)
    return CONFIRMACAO

def pgto_obs(update, context):
    update.message.reply_text('Qual a forma de pagamento?', reply_markup=markup_pgto)
    return FORMAS_PGTO

def forma_pagto(update, context):
    user_data = context.user_data
    forma = update.message.text
    if forma == 'CRÉDITO' or forma == 'DÉBITO':
        user_data['FORMA_PAGTO'] = forma
        user_data['TROCO'] = 0
        update.message.reply_text('Certo! A forma de pagamento escolhida foi {}.'"\n"
            'Pedido enviado ao Chef 👨'.format(forma.lower()))
        return avaliacao_atendimento(update, context)
    else:
        user_data['FORMA_PAGTO'] = forma
        update.message.reply_text('Certo! A forma de pagamento escolhida foi {}.'"\n"
            'Troco para quanto? Digite apenas o valor'.format(forma.lower()))
        return TROCO_PARA


def guarda_troco(update, context):
    user_data = context.user_data
    forma = update.message.text
    user_data['TROCO'] = update.message.text
    try:
        int(user_data['TROCO'])
        if int(user_data['TROCO']) >= float(user_data['VALOR_TOTAL'][3:].replace(',', '.')):
            update.message.reply_text(
            'Tudo bem. Enviaremos troco para {}.'"\n" 'Pedido enviado ao Chef 👨'.format(forma.lower()))
            return avaliacao_atendimento(update, context)
        else:
            update.message.reply_text('O troco informado é menor que o valor da compra, '
                                      'por gentileza me informe corretamente :)')
            return TROCO_PARA
    except:
        update.message.reply_text('Não entendi, digite apenas números. (10, 20, 50) :)')
        return TROCO_PARA

def avaliacao_atendimento(update, context):
```

```
    user_data = context.user_data

    user_data['ID_PEDIDO'] = str(select_bdados(database, "select coalesce(max(id_pedido),0)+1 from pedidos",
                                               username, password))[2:-3]
    user_data['ID_PEDIDO_CLIENTE'] = str(select_bdados(database,"select count(distinct(DATA, HORA))+1 "
                                                       "from pedidos where telegramid = "
                                                + str(user_data['ID_TELEGRAM']),username, password))[2:-3]

    insert = "insert into pedidos (id_pedido, id_pedido_cliente, data, hora, telegramid, id_produto, tipo, " \
             "tipo2, tamanho, nome,quantidade, forma_pagto, observacao, valor_unitario, valor_total, " \
             "valor_compra, troco) select " + str(user_data['ID_PEDIDO']) + "," \
             + str(user_data['ID_PEDIDO_CLIENTE']) + ", '" + str(user_data['DATA_ATENDIMENTO']) \
             + "','" + str(user_data['HORA_ATENDIMENTO']) + "'," + str(user_data['ID_TELEGRAM']) \
             + ", id_produto, tipo, tipo2, tamanho, nome, quantidade, '" + str(user_data['FORMA_PAGTO']) \
             + "','" + str(user_data['OBSERVACAO']) + "',valor_real, valor_real*quantidade,'" \
             + str(user_data['VALOR_TOTAL']) + "', " + str(user_data['TROCO']) \
             + " from carrinho where telegramid =" + str(user_data['ID_TELEGRAM'])

    alteracoes_bdados(database, insert, username, password)
    update.message.reply_text("Posso te pedir uma última coisa? Me diz, de 1 a 5, qual a nota do nosso atendimento?")
    return FIM_PEDIDO

def fim(update, context):
    user_data = context.user_data
    logger.info('alguém notificou nosso atendimento')
    try:
        nota = int(update.message.text)
        if 1 <= nota <= 5:
            user_data['NOTA_ATENDIMENTO'] = nota
            if nota >= 4:
                update.message.reply_text(
                    'Oba, nota {} 😃.' "\n"
                    'Poderia me informar como conseguirei melhorar ainda mais meu atendimento?'.format(nota))
                return MOTIVO_FIM
            else:
                update.message.reply_text(
                    'Apenas {}? 😔' "\n"
                    'Poderia me informar o motivo para que eu melhore meu próximo atendimento?'.format(nota))
                return MOTIVO_FIM
        else:
            update.message.reply_text('Não entendi, digite notas de 1 a 5,por exemplo 5 :D')
            return FIM_PEDIDO
    except:
        update.message.reply_text('Não entendi, digite apenas números, por exemplo 5 :D')
        return FIM_PEDIDO

def set_timer(update, context):
    count = 0
    if count == 0:
        chat_id = update.message.chat_id
        try:
            due = int(15)
            if due < 0:
                update.message.reply_text('Verificar!')
                return
            if 'job' in context.chat_data:
                old_job = context.chat_data['job']
                old_job.schedule_removal()
            new_job = context.job_queue.run_once(alarm, due, context=chat_id)
            context.chat_data['job'] = new_job
            count += 1
            update.message.reply_text('Timer iniciado!')
            count += 1
            return  NOTA_PIZZA
        except (IndexError, ValueError):
            count += 1
            update.message.reply_text('Usage: /set <seconds>')
    else:
        count += 1
        return final_final(update, context)


def alarm(context):
    job = context.job
    context.bot.send_message(job.context,
                             text="Oi, tudo bem? Depois de comer a pizza, "
                                  "qual nota daria para ela? digite um número de 1 a 5")
    return NOTA_PIZZA

def clean_text(text):
    text = text.lower() # Leave all lowercase characters
    text = normalize('NFKD', text).encode('ASCII', 'ignore').decode('ASCII')
    text = [re.sub(r'^https?:\/\/.[\r\n]', ' ', word) for word in text.split(" ")]
    text = " ".join(text) # Uniting text again separated by space
    text = re.sub(r'<[^>]+>',' ',text) # hange the '<br />' section to space
    text = re.sub(r'[^a-z]', ' ', text) # Changes scores and numbers to space
```

```
    text = re.sub(r"\s+[a-z]\s+", ' ', text) # Change single characters to space
    text = re.sub(r'([a-z])(?=\1{2,})', "", text) # Removes repeated characters over 2 times
    text = re.sub(r'[ ]{1,}',' ',text) # Removes duplicate spaces in the string
    text = text.strip() # Remove spaces at the beginning and end of the string

    # Removes the words found in the stop_sword list
    text = [word for word in text.split(" ") if not word in stop_words]
    text = " ".join(text) # Join text again separated by space
    return text

def nlp(update, context):
    user_data = context.user_data
    user_data['AVALIACAO_ATENDIMENTO'] = update.message.text
    user_data['NLP_ATENDIMENTO'] = str(
        preditor_avaliacao.predict(bow.transform([clean_text(user_data['AVALIACAO_ATENDIMENTO'])])))
    update.message.reply_text("Anotado, avaliaremos junto a equipe sua avaliação")

    user_data['ID_PEDIDO_CLIENTE'] = str(
        select_bdados(database,"select count(distinct(DATA, HORA)) from pedidos where telegramid = "+ str(
            user_data['ID_TELEGRAM']),username, password))[2:-3]

    insert = "insert into avaliacoes(id_pedido, id_pedido_cliente, telegramid, nota_atendimento, avaliacao_" \
             "atendimento, nlp_atendimento) " \
        "select id_pedido, id_pedido_cliente, telegramid," + str(user_data['NOTA_ATENDIMENTO']) \
            + ", '" + str(user_data['AVALIACAO_ATENDIMENTO']) + \
        "', '" +str(user_data['NLP_ATENDIMENTO'][2:-2]) + "' from pedidos where " \
        "telegramid = " + str(user_data['ID_TELEGRAM']) + " and id_pedido_cliente = " \
            + str(user_data['ID_PEDIDO_CLIENTE'])

    alteracoes_bdados(database, insert, username, password)
    return finalizar(update, context)


def nota_pizza(update, context):
    user_data = context.user_data
    try:
        nota = int(update.message.text)
        if 1 <= nota <= 5:
            user_data['NOTA_PIZZA'] = nota
            update.message.reply_text("Certo, poderia me descrever o motivo desta nota?".format(nota))
            return TESTE
        else:
            update.message.reply_text('Não entendi, digite notas de 1 a 5,por exemplo 5 :D')
            return NOTA_PIZZA
    except:
        update.message.reply_text('Não entendi, digite apenas números, por exemplo 5 :D')
        return NOTA_PIZZA

def nlp2(update, context):
    user_data = context.user_data
    user_data['AVALIACAO_PIZZA'] = update.message.text
    user_data['NLP_PIZZA'] = str(preditor_avaliacao.predict(bow.transform([user_data['AVALIACAO_PIZZA']])))
    user_data['ID_PEDIDO_CLIENTE'] = str(
        select_bdados(database, "select count(distinct(DATA, HORA)) from pedidos where telegramid = "
                      + str(user_data['ID_TELEGRAM']), username, password))[2:-3]

    update_tbl = "update avaliacoes set nota_pizza = "+ str(user_data['NOTA_PIZZA']) \
                 + ", avaliacao_pizza = '" + str(user_data['AVALIACAO_PIZZA']) + \
        "', nlp_pizza = '" + str(user_data['NLP_PIZZA'][2:-2]) + "' where telegramid = " \
                 + str(user_data['ID_TELEGRAM']) + " and id_pedido_cliente = " + str(user_data['ID_PEDIDO_CLIENTE'])

    update_bdados(database, update_tbl, username, password)
    update.message.reply_text("Muito obrigado pelo seu feedback 😃")
    return final_final(update, context)

def finalizar(update, context):
    logger.info('alguém finalizou o atendimento')
    update.message.reply_text("A Chatizza agradece seu contato."'\n' "Até a próxima 🙋")
    return set_timer(update, context)

def final_final(update, context):
    update.message.reply_text("A Chatizza agradece seu contato."'\n' "Até a próxima 🙋")
    return ConversationHandler.END

def timeout(update, context):
    logger.info('alguém parou de responder nosso atendimento')
    update.effective_message.reply_text("Bem, vou finalizar nossa conversa por falta de interação ok?" '\n'
                                        "Caso queira iniciar um novo atendimento, estamos à disposição.")
    return ConversationHandler.END

# Bot
def main():
    print('Aplicação em execução 🚀')

    updater = Updater(token, use_context=True)
```

```
# Creating the dispatcher to register handlers
dp = updater.dispatcher

criando_bdados(database, username, password)

start_filter = StartFilter()
end_filter = EndFilter()

# Button display
conv_handler = ConversationHandler(
    entry_points=[CommandHandler('start', start),
                  MessageHandler(Filters.text & start_filter, start)],

    states={

        # Register
        CAD_NOME: [MessageHandler(Filters.text,
                                  armazena_nome)
                   ],

        CAD_CPF: [MessageHandler(Filters.text,
                                 armazena_cpf)
                  ],

        CAD_CEP: [MessageHandler(Filters.text,
                                 armazena_cep)
                  ],

        VALID_RUA:[MessageHandler(Filters.regex('^(SIM)$'),
                                  pass_numero),
                   MessageHandler(Filters.regex('^(NÃO)$'),
                                  troca_rua)
                   ],
        DIGITA_NUMERO:[MessageHandler(Filters.text,
                                  armazena_numero)
                     ],

        TROCA_RUA:[MessageHandler(Filters.text,
                                  guarda_rua)
                ],

        DIGITA_COMPLEMENTO:[MessageHandler(Filters.text,
                                  armazena_complemento)

                     ],

        CAD_GENERO: [MessageHandler(Filters.text,
                                  armazena_genero)
                    ],

        CAD_IDADE: [MessageHandler(Filters.text,
                                 armazena_idade)
                    ],

        CAD_TEL: [MessageHandler(Filters.text,
                                 armazena_tel)
                  ],

        # Post registration interactions
        PRIM_INTERACAO: [MessageHandler(Filters.regex('^(QUANTAS 🤔)$'),
                                  dimensionamento_choice),
                         MessageHandler(Filters.regex('^(CARDÁPIO 📖)$'),
                                  cardapio_choice),
                         MessageHandler(Filters.regex('^(INFORMAÇÕES 💡)$'),
                                  infos_choice),
                         MessageHandler(Filters.regex('^(CADASTRO 📝)$'),
                                 altera_cadastro),
                         MessageHandler(Filters.regex('^(FINALIZAR 🙅)$'),
                                  final_final)
                         ],

        # Calculator interactions
        PRIM_ITERACAO_DIGITE: [MessageHandler(Filters.text,
                                        dimensionamento_homem),
                               ],

        PRIM_ITERACAO_DIGITE2: [MessageHandler(Filters.text,
                                         dimensionamento_mulher),
                                ],

        PRIM_ITERACAO_DIGITE3: [MessageHandler(Filters.text,
                                         dimensionamento_crianca),
                                ],
```

```
# Menu interactions
CARDAPIO_OPCOES: [MessageHandler(Filters.regex('^(TRADICIONAIS 🍕)$'),
                                 tamanhos),
                  MessageHandler(Filters.regex('^(DOCES 🍫)$'),
                                 opcoes_doces),
                  MessageHandler(Filters.regex('(^BEBIDAS 🥤)'),
                                 opcoes_bebidas)
                  ],

TAMANHO: [MessageHandler(Filters.regex('^(GRANDE - 8 PEDAÇOS)$'),
                         sabores),
          MessageHandler(Filters.regex('^(BROTO - 4 PEDAÇOS)$'),
                         sabores)
          ],

NUM_SABORES: [MessageHandler(Filters.regex('^(1 SABOR)$'),
                             opcoes_pizzas),
              MessageHandler(Filters.regex('^(2 SABORES)$'),
                             opcoes_pizzas)
              ],


# Interactions to add or finalize order
ADICIONAR: [MessageHandler(Filters.regex('^(ADICIONAR ☝)$'),
                           cardapio_choice),
            MessageHandler(Filters.regex('^(REMOVER ❌)'),
                           remove_item),
            MessageHandler(Filters.regex('^(FINALIZAR PEDIDO ✅)'),
                           fim_pedido)
            ],

FINALIZAR_COMPRA:[MessageHandler(Filters.regex('^(ADICIONAR ☝)$'),
                           cardapio_choice),
            MessageHandler(Filters.regex('^(REMOVER ❌)'),
                           remove_item),
            MessageHandler(Filters.regex('^(CONFIRMAR COMPRA ✅)'),
                           observacao)
            ],

OBSERVACAO: [MessageHandler(Filters.text,
                            salva_obs)
             ],

CONFIRMACAO:[MessageHandler(Filters.regex('^(SIM!)$'),
                            pgto_obs),
                MessageHandler(Filters.regex('^(NÃO!)$'),
                            troca_rua)
            ],

FORMAS_PGTO: [MessageHandler(Filters.regex('^(CRÉDITO)$'),
                           forma_pagto),

               MessageHandler(Filters.regex('^(DÉBITO)$'),
                           forma_pagto),

               MessageHandler(Filters.regex('^(DINHEIRO)'),
                           forma_pagto)
               ],

TROCO_PARA: [MessageHandler(Filters.text,
                            guarda_troco)
               ],


FIM_PEDIDO: [MessageHandler(Filters.text,
                            fim),
             ],

QTD_PEDIDO: [MessageHandler(Filters.text,
                           quantidades),
            ],

ADD_ITEM: [MessageHandler(Filters.text,
                            add_item),
              ],

NOTA_PIZZA: [MessageHandler(Filters.text,
                          nota_pizza),
           ],

# Interactions for text identification
MOTIVO_FIM: [MessageHandler(Filters.text,
                            nlp)
```

```
                    ],

            TESTE: [MessageHandler(Filters.text,
                                   nlp2)],
                    ConversationHandler.TIMEOUT:[MessageHandler(Filters.text,
                                    timeout)
                    ],

            ALTERAR_CAD: [MessageHandler(Filters.regex('^(NOME)$'),
                                         forma_pagto),
                          MessageHandler(Filters.regex('^(CPF)$'),
                                         forma_pagto),
                          MessageHandler(Filters.regex('^(NASCIMENTO)'),
                                         forma_pagto),
                          MessageHandler(Filters.regex('^(CEP)'),
                                         forma_pagto),
                          MessageHandler(Filters.regex('^(RUA)'),
                                         forma_pagto),
                          MessageHandler(Filters.regex('^(NUMERO)'),
                                         forma_pagto),
                          MessageHandler(Filters.regex('^(COMPLEMENTO)'),
                                         forma_pagto),
                          MessageHandler(Filters.regex('^(TELEFONE)'),
                                         forma_pagto),
                          MessageHandler(Filters.regex('^(GENERO)'),
                                         forma_pagto),
                          ],
        },

        fallbacks=[MessageHandler(Filters.text & end_filter, final_final)],

        conversation_timeout=40000

    )

    # Handler for post-order timer
    dp.add_handler(CommandHandler('set',set_timer))

    # Button Handler
    dp.add_handler(conv_handler)

    # Error log
    dp.add_error_handler(error)
    updater.start_polling()
    updater.idle()

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