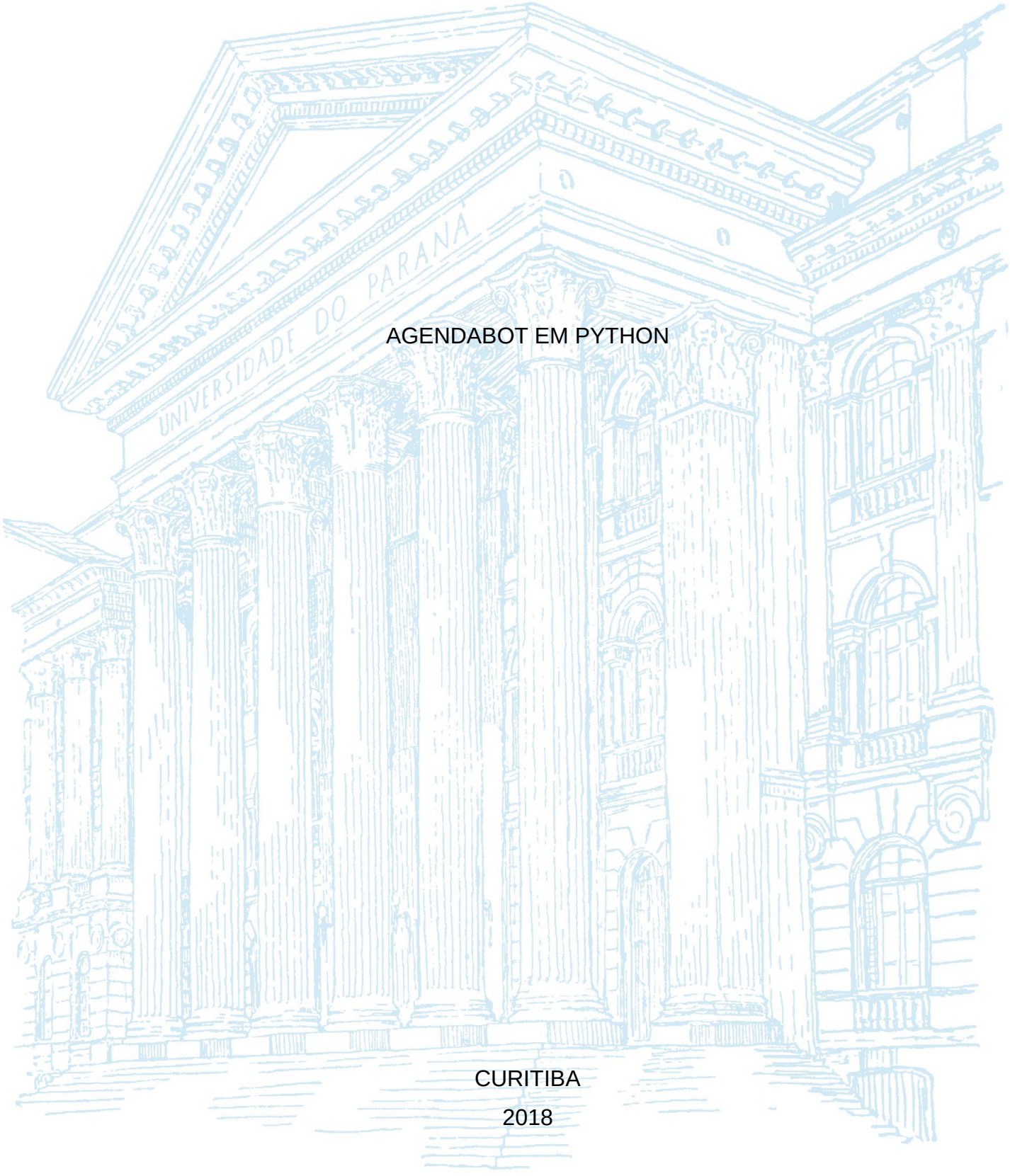
UNIVERSIDADE FEDERAL DO PARANÁ

CRISTINA CHEN

LUCAS GABRIEL NADOLNY



AGENDABOT EM PYTHON

CURITIBA

2018

CRISTINA CHEN (GRR20185643)
LUCAS GABRIEL NADOLNY (GRR20185670)

AGENDABOT EM PYTHON

Relatório apresentado à disciplina Fundamentos de Programação de Computadores do Curso de Graduação em Matemática da Universidade Federal do Paraná.

Orientador: Prof. Jackson Antônio do Prado Lima

CURITIBA
2018

SUMÁRIO

1 INTRODUÇÃO

Esse relatório apresenta informações relativas ao trabalho realizado em Python acerca do Bot no Telegram, realizado em dupla.

A linguagem Python foi concebida em 1989 pelo holandês Guido van Rossum, visto que ele estava desenvolvendo a linguagem ABC no CWI em Amsterdã – Holanda, e estava encontrando deficiências nessa linguagem. Tentando suprir esses problemas visto com o ABC, o holandês criou o Python com base em C.

O Telegram foi fundado em 2013 pelos irmãos Nikolai e Pavel Durov, é um aplicativo concorrente do WhatsApp. Os usuários podem enviar mensagens e trocar fotos, vídeos, stickers e arquivos de qualquer tipo. O Telegram também possui criptografia ponta-a-ponta opcional. Os clientes do Telegram possuem código aberto, porém seus servidores são proprietários. O serviço também providencia APIs para desenvolvedores independentes.

2 OBJETIVOS

2.1 Objetivo geral

O objetivo é criar um bot para o Telegram que funcione como uma agenda. O bot tem os seguintes comandos:

- Criar evento: pede o título do evento; a data; o horário; os detalhes e pergunta se quer adicionar um lembrete;
- Lista de eventos: retorna os eventos registrados e com esses, é possível editá-los;
- Cancelar alguma ação;
- Deletar eventos;

3 DESENVOLVIMENTO

3.1 O trabalho

O programa consiste em 25 funções afim de serem executados pelo Bot do Telegram. Podendo também, mostrar mensagens no console, caso o programador queira, isso é o que a função ‘listener’ faz.

```
def listener(mensagens):
    for mensagem in mensagens:
        if mensagem.content_type == 'text':
            print(str(mensagem.chat.first_name) + " [" + str(mensagem.chat.id) + "]: " + mensagem.text)
```

Figura 1.1- Função que mostra na tela do console as mensagens enviadas pelo usuário

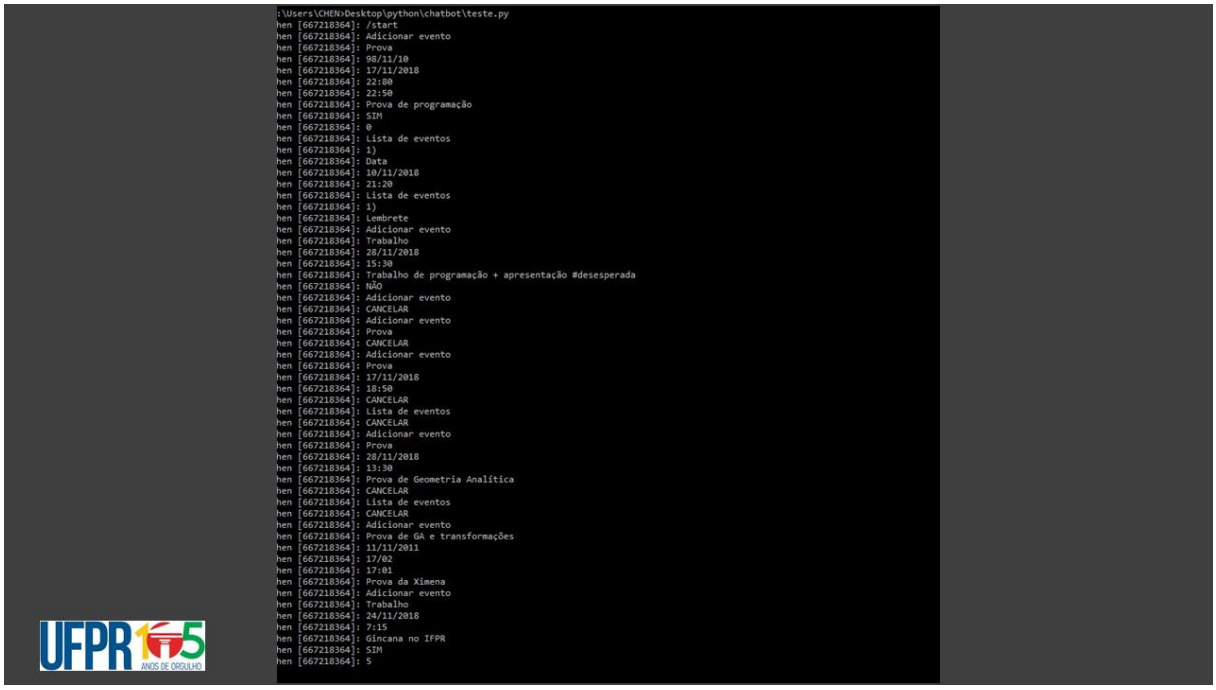


Figura 1.2- Imagem ilustrativa de como ocorre no console

3.2 Funções do arquivo ‘funções’

A função ‘dois_digitos’ foi feita pois, ao importar o datetime, estava tendo problemas em imprimir o horário, uma vez que o datetime somente executa 1 dígito, e para o horário, executar apenas um dígito não ficava elegante. Exemplo: 10:05 ficava 10:5;

```python
def dois_digitos(n):
    if n < 10:
        return "0"+str(n)
    else:
        return str(n)
```

Figura 1.1- Função para deixar a impressão do horário mais elegante

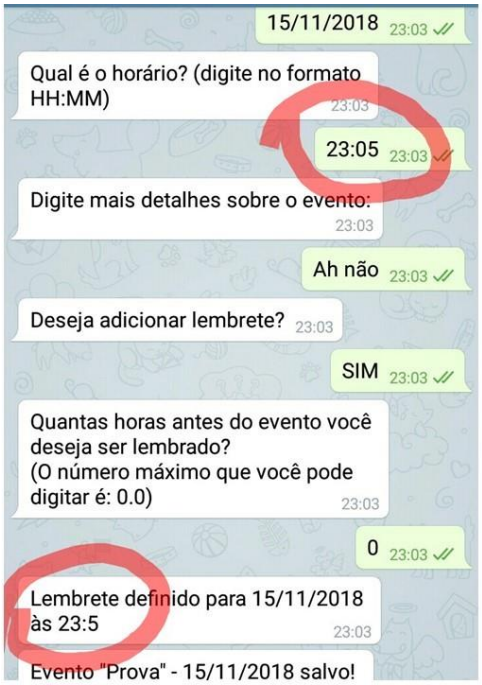


Figura 1.2- Amostra da imagem de como resultava o horário sem essa função

A função ‘validar_data’ foi feita para validar a data, muito importante, uma vez em que em nosso AgendaBot tem a função do lembrete; se os valores digitados não corresponderem a uma data válida, exemplo se o mês for 13 ou ou dia 32, quando é chamada a função date do datetime ocorrerá ValueError nesse caso é retornado falso, se conseguir chamar a função é retornado verdadeiro.

```python
def validar_data(dia, mes, ano):
    try:
        datetime.date(ano, mes, dia)
        return True
    except ValueError:
        return False
```

Figura 2- Função para validar a data

A função ‘texto_para_datetime’ converte o texto no formato "DD/MM/AAAA HH:MM" para um datetime.

```python
def texto_para_datetime(texto):

    dataT = texto.split()
    data = [int(n) for n in dataT[0].split("/")]
    hora = [int(n) for n in dataT[1].split(":")]
    return datetime.datetime(data[-1], data[-2], data[-3], hora[0], hora[1])
```

Figura 3- Função para converter o texto para datetime

A função ‘datetime_para_texto’ converte um datetime para um texto no formato "DD/MM/AAAA HH:MM".

```python
def datetime_para_texto(dt):
    return "{}/{}/{} {}:{}".format(dois_digitos(dt.day), dois_digitos(dt.month),
     dt.year, dois_digitos(dt.hour), dois_digitos(dt.minute))
```

Figura 4- Função para converter datetime para texto

A função ‘salvar_edicao’ salva uma edição feita no dicionário para o arquivo.

```python
def salvar_edicao(eventos, cid):
    linhas_salvar = []
    for linha in eventos[cid]:
        linhas_salvar.append(linha[0]+"\n")
        linhas_salvar.append(datetime_para_texto(linha[1])+"\n")
        linhas_salvar.append(linha[2]+"\n")
        if linha[3] == 0:
            linhas_salvar.append("0\n")
        else:
            linhas_salvar.append(datetime_para_texto(linha[3])+"\n")
    with open('eventos/{}.txt'.format(cid), 'w') as f:
        f.writelines(linhas_salvar)
```

Figura 5- Função para salvar uma alteração no dicionário para o arquivo

3.3 Adicionar evento

A função ‘comando_start’ envia uma mensagem de boas-vindas para o usuário novo e em seguida pede para escolher uma das opções ‘adicionar evento’ ou ‘ lista de eventos’.

```python
@bot.message_handler(commands=['start'])
def comando_start(mensagem):
    cid = mensagem.chat.id
    if cid not in usuariosConhecidos:
        usuariosConhecidos.append(cid)
        lembretes[cid] = {}
        eventos[cid] = []
        with open('usuarios.txt', 'a') as f:
            f.write(str(cid)+"\n")
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        bot.send_message(cid, "Bem vindo ao AgendaBot, gostaria de adicionar algum evento?", reply_markup=selecionarComando)
    else:
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
```

Figura 1.1- Função feita para novos usuários

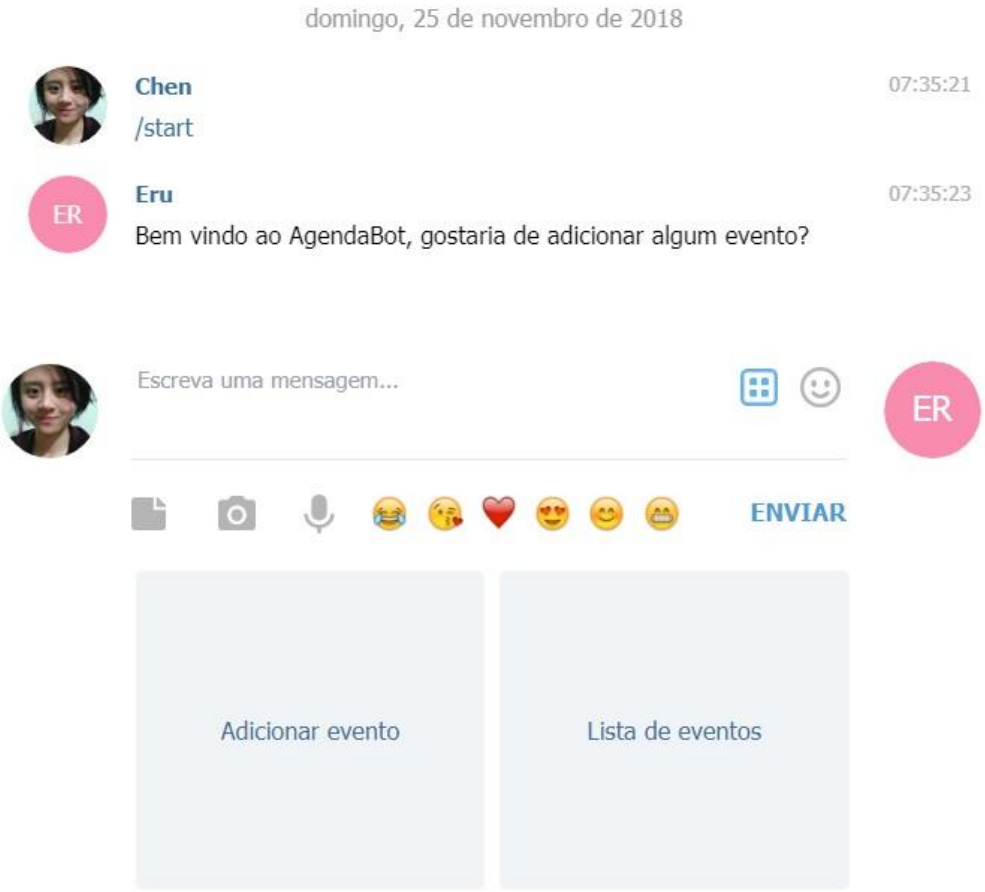


Figura 1.2- Amostra da imagem do que ocorre quando o usuário insere o comando ‘/start’

A função ‘passo_do_usuario’ serve para evitar erros que aconteceriam se tentasse incluir o passo de um usuário que não está no dicionário além de salvar as coisas em arquivos, é criado também, o dicionário para lembrete e a lista que vai ser a matriz de eventos.

```python
def passo_do_usuario(uid):
    if uid in passoUsuario:
        return passoUsuario[uid]
    else:
        usuariosConhecidos.append(uid)
        lembretes[uid] = {}
        eventos[uid] = []
        f = open('usuarios.txt', 'a')
        f.write(str(uid)+"\n")
        f.close()

        try:
            f = open('linhatemp/{}.txt'.format(uid), 'x')
            f.close()
        except FileExistsError:
            pass
        try:
            f = open('eventos/{}.txt'.format(uid), 'x')
            f.close()
        except FileExistsError:
            pass

        passoUsuario[uid] = 0
        v = open('passo/{}.txt'.format(uid), 'w')
        v.write("0")
        v.close()
        return 0
```

Figura 2- Função para evitar ‘erros’

A função ‘adicionar_evento’ é chamada quando o usuário seleciona a opção ‘Adicionar evento’ e então o bot manda uma mensagem pedindo o título do evento.

```python
@bot.message_handler(func=lambda message: message.text == comandos[0] and passo_do_usuario(message.chat.id) == 0)
def adicionar_evento(mensagem):
    cid = mensagem.chat.id
    bot.send_message(cid, "Digite um título, por favor", reply_markup=cancelar)
    passoUsuario[cid] = 1
    with open('passo/{}.txt'.format(cid), 'w') as f:
        f.write("1")
```

Figura 3.1- Função para adicionar um título ao evento

Figura 3.2- Amostra de como acontece no bot do Telegram

A função ‘adicionar_evento1’ serve para cancelar o ato anterior ou salvar o título e perguntar a data.

```python
@bot.message_handler(func=lambda message: passo_do_usuario(message.chat.id) == 1)
def adicionar_evento1(mensagem):
    cid = mensagem.chat.id
    if mensagem.text == "CANCELAR":
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
    else:
        linhatemp[str(cid)+"titulo"] = mensagem.text
        with open('linhatemp/{}.txt'.format(cid), 'a') as f:
            f.write(mensagem.text+'\n')
        bot.send_message(cid, "Qual é a data? (digite no formato DD/MM/AAAA)", reply_markup=cancelar)
        passoUsuario[cid] = 2
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("2")
```

Figura 4.1- Função feita para organizar o passo, este em questão, está no passo1

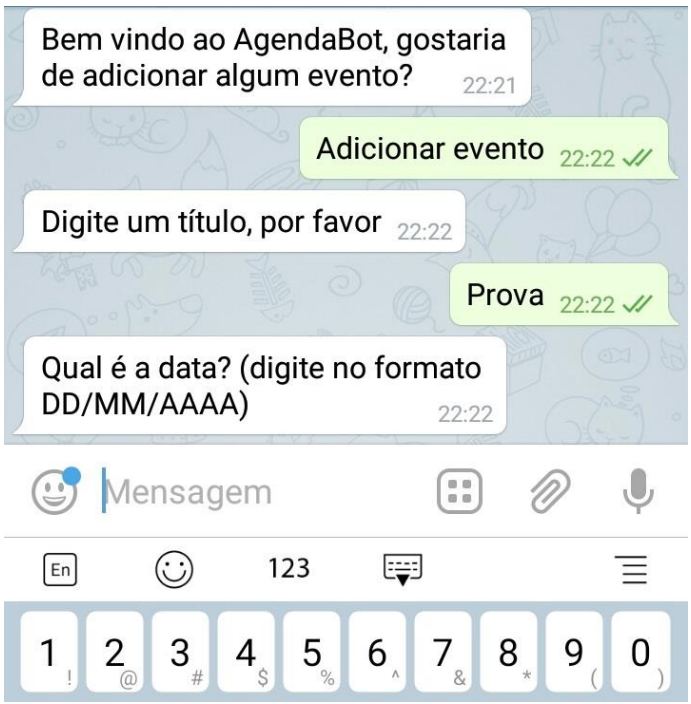


Figura 4.2- Figura ilustrativa de como é executado no bot

A função ‘adicionar_evento2’ serve para cancelar os atos anteriores ou salvar a data e perguntar o horário (validando-a).

```python
@bot.message_handler(func=lambda message: passo_do_usuario(message.chat.id) == 2)
def adicionar_evento2(mensagem):
    cid = mensagem.chat.id
    if mensagem.text == "CANCELAR":
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        del linhatemp[str(cid)+"titulo"]
        f = open('linhatemp/{}.txt'.format(cid), 'w')
        f.writelines([""])
        f.close()
        bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
    else:
        try:
            data = mensagem.text
            data = [int(n) for n in data.split("/")]
            if funcoes.validar_data(data[0], data[1], data[2]) == False or len(data) != 3:
                raise ValueError("Data inválida")
            linhatemp[str(cid)+"data"] = datetime.datetime(data[-1], data[-2], data[-3])
            with open('linhatemp/{}.txt'.format(cid), 'a') as f:
                f.write(funcoes.datetime_para_texto(linhatemp[str(cid)+"data"]))
            bot.send_message(cid, "Qual é o horário? (digite no formato HH:MM)", reply_markup=cancelar)
            passoUsuario[cid] = 3
            with open('passo/{}.txt'.format(cid), 'w') as v:
                v.write("3")
        except (ValueError, IndexError):
            bot.send_message(cid, """Data inválida! Digite a data no formato DD/MM/AAAA !\nExemplo:
                 dia 17 de janeiro de 2029 corresponde a data 17/01/2029.""", reply_markup=cancelar)
```

Figura 5.1- Função feita para o passo 2

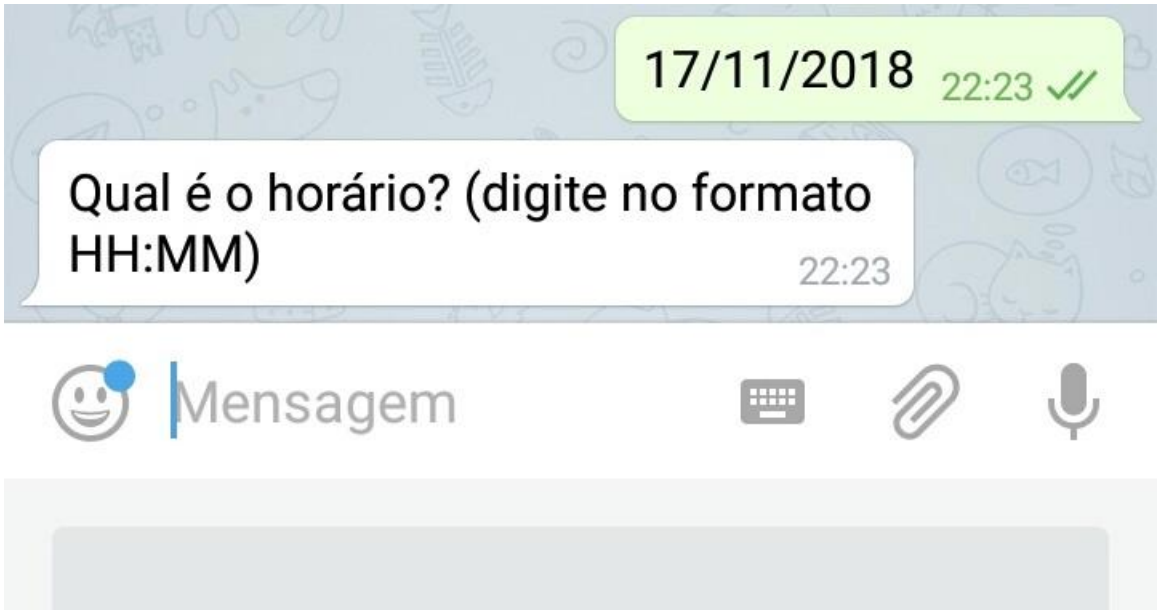


Figura 5.2- Figura ilustrativa de como o Bot reage após receber a data

A função ‘adicionar_evento3’ serve para cancelar os atos anteriores ou validar o horário e perguntar por detalhes.

```python
@bot.message_handler(func=lambda message: passo_do_usuario(message.chat.id) == 3)
def adicionar_evento3(mensagem):
    cid = mensagem.chat.id
    if mensagem.text == "CANCELAR":
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        del linhatemp[str(cid)+"titulo"]
        del linhatemp[str(cid)+"data"]
        with open('linhatemp/{}.txt'.format(cid), 'w') as f:
            f.writelines([''])
        bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
    else:
        try:
            horario = mensagem.text
            horario = [int(n) for n in horario.split(":")]
            if horario[0] < 0 or horario[0] > 23:
                raise ValueError("Hora inválida")
            if horario[1] < 0 or horario[1] > 59:
                raise ValueError("Minuto inválido")
            if len(horario) != 2:
                raise ValueError("Horário inválido")
            linhatemp[str(cid)+"data"] = linhatemp[str(cid)+"data"].replace(hour=horario[0], minute=horario[1])
            with open('linhatemp/{}.txt'.format(cid), 'w') as f:
                f.write(linhatemp[str(cid)+"titulo"]+"\n"+funcoes.datetime_para_texto(linhatemp[str(cid)+"data"])+"\n")
            bot.send_message(cid, "Digite mais detalhes sobre o evento:", reply_markup=cancelar)
            passoUsuario[cid] = 4
            with open('passo/{}.txt'.format(cid), 'w') as v:
                v.write("4")
        except (ValueError, IndexError):
            bot.send_message(cid, "Horário inválido! Digite o horário no formato HH:MM !\nExemplo: duas e quarenta e cinco da tarde são 14:45")
```

Figura 6.1- Função feita para o passo 3

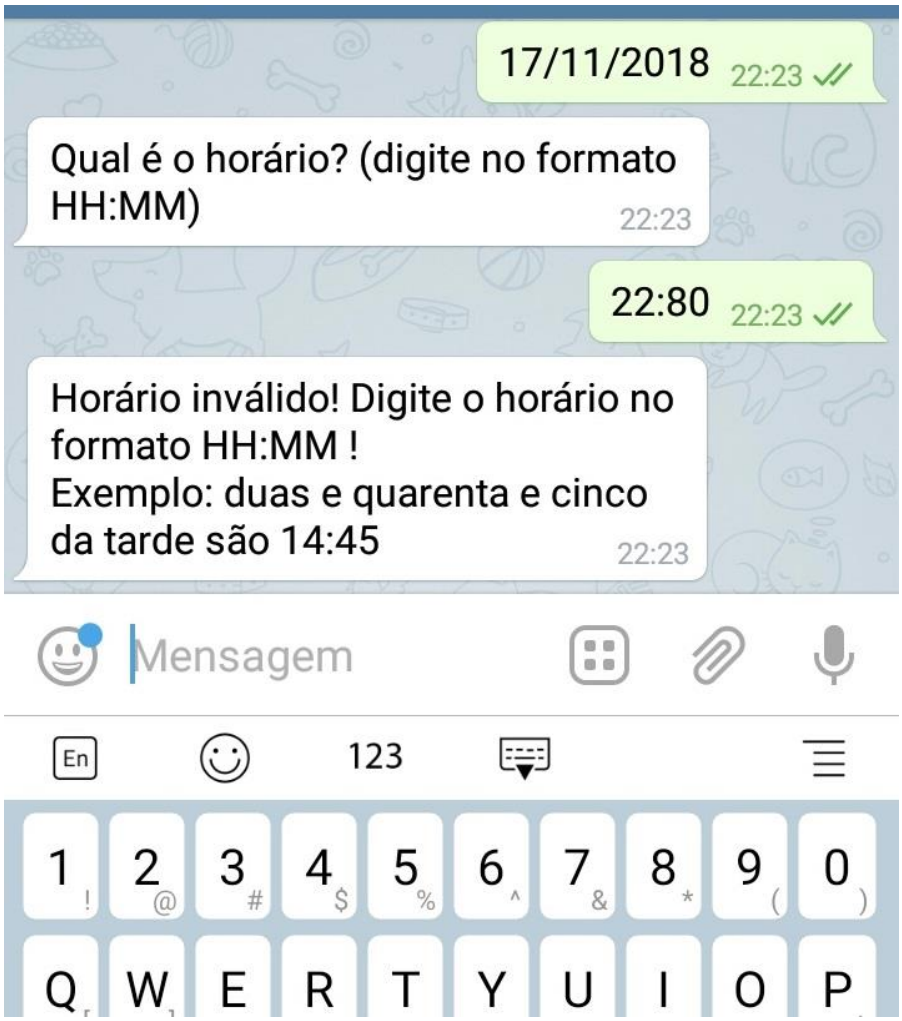


Figura 6.2- Foto ilustrativa de como o bot reage ao digitar um horário inválido

A função ‘adicionar_evento4’ serve para cancelar todos os atos anteriores ou salvar os detalhes e perguntar se o usuário quer um lembrete.

```python
@bot.message_handler(func=lambda message: passo_do_usuario(message.chat.id) == 4)
def adicionar_evento4(mensagem):
    cid = mensagem.chat.id
    if mensagem.text == "CANCELAR":
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        del linhatemp[str(cid)+"titulo"]
        del linhatemp[str(cid)+"data"]
        with open('linhatemp/{}.txt'.format(cid), 'w') as f:
            f.writelines([''])
        bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
    else:
        linhatemp[str(cid)+"detalhes"] = mensagem.text
        with open('linhatemp/{}.txt'.format(cid), 'a') as f:
            f.write(mensagem.text+"\n")
        agora = datetime.datetime.now()
        if agora < linhatemp[str(cid)+"data"]:
            bot.send_message(cid, "Deseja adicionar lembrete?", reply_markup=sim_ou_não)
            passoUsuario[cid] = 5
            with open('passo/{}.txt'.format(cid), 'w') as v:
                v.write("5")
        else:
            salvar_evento(eventos, cid, linhatemp[str(cid)+"titulo"], linhatemp[str(cid)+"data"], linhatemp[str(cid)+"detalhes"], 0, selecionarComando)
            passoUsuario[cid] = 0
            with open('passo/{}.txt'.format(cid), 'w') as v:
                v.write("0")
            del linhatemp[str(cid)+"titulo"]
            del linhatemp[str(cid)+"data"]
            del linhatemp[str(cid)+"detalhes"]
            with open('linhatemp/{}.txt'.format(cid), 'w') as f:
                f.writelines([''])
```

Figura 7.1- Função feita para o passo 4

Figura 7.2- Imagem ilustrativa para adicionar lembrete

A função ‘salvar_evento’ salva no dicionário de eventos e no arquivo do usuário cid, o seu evento e organiza os eventos do dicionário por ordem de data. O bot vai tentar colocar as informações dentro da matriz de eventos (try), mas se no dicionário nada estiver salvo no cid, vai dar KeyError, nesse caso é criada uma matriz com o evento dentro. Depois é organizado de dois a dois colocando em ordem crescente de tempo

e então vai de dois em dois até o final, depois até o final menos 1, depois menos 2 e assim por diante; se a data que está na posição i, for antes da que está na i-1, então a função as inverte.

```
def salvar_evento(eventos, cid, titulo, data, detalhes, lembrete, selecionarComando):
    try:
        eventos[cid].append([titulo, data, detalhes, lembrete])
    except KeyError:#2
        eventos[cid] = [[titulo, data, detalhes, lembrete]]

    if lembrete == 0:
        lembreteTexto = "0"
    else:
        lembreteTexto = funcoes.datetime_para_texto(lembrete)

    f = open('eventos/{}.txt'.format(cid), 'a')
    f.write(str(titulo)+"\n"+funcoes.datetime_para_texto(data)+"\n"+str(detalhes)+"\n"+lembreteTexto+"\n")
    f.close()

    for k in range(len(eventos[cid])):
        for i in range(1, len(eventos[cid]) - k):
            if eventos[cid][i][1] < eventos[cid][i-1][1]:
                eventos[cid][i], eventos[cid][i-1] = eventos[cid][i-1], eventos[cid][i]

    bot.send_message(cid, "Evento \"{}\" - {} salvo!\nO que deseja fazer?".format(titulo, funcoes.datetime_para_texto(data)), reply_markup=selecionarComando)
```

Figura 8- Função que guarda os eventos que o usuário deseja salvar

A função ‘enviar_lembrete’ serve para enviar a mensagem de lembrete para o usuário.

```
def enviar_lembrete(usuario, titulo, data):
    bot.send_message(usuario, "Você terá \"{}\" em {}".format(titulo, data))
```

Figura 9.1- Função que envia um lembrete para o usuário

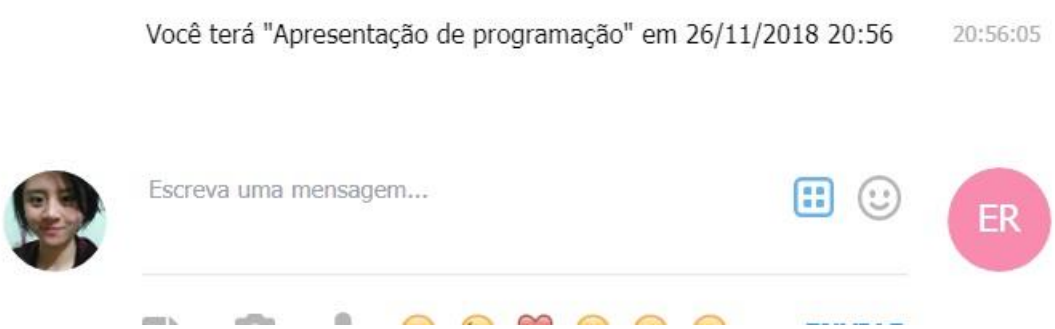


Figura 9.2- Imagem ilustrativa para o bot enviar um lembrete

A ‘função’ sched organiza todos os lembretes em uma fila crescente e programa o envio dos lembretes

```
fila = sched.scheduler(time.time, time.sleep)
def enviar_lembrete(usuario, titulo, data):
    bot.send_message(usuario, "Você terá \"{}\" em {}".format(titulo, data))

for cid in usuariosConhecidos:
    agora = datetime.datetime.now()
    for i in range(len(eventos[cid])):
        if eventos[cid][i][-1] != 0:
            if eventos[cid][i][-1] < agora:
                eventos[cid][i][-1] = 0
            else:
                st = (eventos[cid][i][-1] - agora).total_seconds()
                lembretes[cid][eventos[cid][i][0]+funcoes.datetime_para_texto(eventos[cid][i][1])] =
                fila.enter(st, 1, enviar_lembrete, argument=(cid, eventos[cid][i][0], funcoes.datetime_para_texto(eventos[cid][i][1])))
t = threading.Thread(target=fila.run)
t.start()
```

Figura 9.3- Função sched ajuda a organizar todos os eventos em uma linha crescente de horário.

A função ‘lembrete’ permite digitar em quantos dias/horas de antecedência o usuário gostaria de ser lembrado para o evento informado, caso o evento seja posterior a data de agora. Se o evento for antecedente à data de agora mostrará uma mensagem para o usuário de que não é mais possível criar um lembrete. Se, por ventura, o usuário não quiser um lembrete, então o bot salvará todas as outras informações e perguntará o que o usuário deseja fazer em sequência. Mais ainda, se o usuário não inserir ‘sim’ ou ‘não’ então o programa irá enviar uma mensagem pedindo para o usuário escolher uma das opções.

```
@bot.message_handler(func=lambda message: passo_do_usuario(message.chat.id) == 5)
def lembrete(mensagem):
    cid = mensagem.chat.id
    if mensagem.text == "SIM":
        agora = datetime.datetime.now()
        if (linhatemp[str(cid)+"data"]-agora).days >= 1:
            bot.send_message(cid, "Quantos dias antes do evento você deseja ser lembrado?\n(O número máximo que você pode digitar é: {})".
                                                format((linhatemp[str(cid)+"data"]-agora).days), reply_markup=esconderTeclado)
            linhatemp[str(cid)+"casolembrete"] = 1
            with open('linhatemp/{}.txt'.format(cid), 'a') as f:
                f.write("1")
            passoUsuario[cid] = 6
        elif linhatemp[str(cid)+"data"] < agora:
            bot.send_message(cid, "Não é mais possível criar um lembrete.")
            salvar_evento(eventos, cid, linhatemp[str(cid)+"titulo"], linhatemp[str(cid)+"data"], linhatemp[str(cid)+"detalhes"], 0, selecionarComando)
            passoUsuario[cid] = 0
            with open('passo/{}.txt'.format(cid), 'w') as v:
                v.write("0")
            del linhatemp[str(cid)+"titulo"]
            del linhatemp[str(cid)+"data"]
            del linhatemp[str(cid)+"detalhes"]
        else:
            bot.send_message(cid, "Quantos horas antes do evento você deseja ser lembrado?\n(O número máximo que você pode digitar é: {})".
                format((linhatemp[str(cid)+"data"]-agora).total_seconds()//3600) , reply_markup=esconderTeclado)
            linhatemp[str(cid)+"casolembrete"] = 2
            with open('linhatemp/{}.txt'.format(cid), 'a') as f:
                f.write("1")
            passoUsuario[cid] = 6
            with open('passo/{}.txt'.format(cid), 'w') as v:
                v.write("6")
    elif mensagem.text == "NÃO":
        salvar_evento(eventos, cid, linhatemp[str(cid)+"titulo"], linhatemp[str(cid)+"data"], linhatemp[str(cid)+"detalhes"], 0, selecionarComando)
        del linhatemp[str(cid)+"titulo"]
        del linhatemp[str(cid)+"data"]
        del linhatemp[str(cid)+"detalhes"]
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
    else:
        bot.send_message(cid, "Por favor selecione uma das opções", reply_markup=sim_ou_não)
```

Figura 10.1- Função feita para o passo 5

Figura 10.2- Imagem ilustrativa para o passo5

A função ‘lembrete1’ serve para validar o evento, caso o lembrete seja antecedente a agora, o usuário receberá uma mensagem contendo ‘Não é mais possível criar um lembrete’ e então não será criado um lembrete.

```
@bot.message_handler(func=Lambda message: passo_do_usuario(message.chat.id) == 6)
def lembrete1(mensagem):
    cid = mensagem.chat.id
    agora = datetime.datetime.now()
    if linhatemp[str(cid)+"data"] < agora:
        bot.send_message(cid, "Não é mais possível criar um lembrete.")
        salvar_evento(eventos, cid, linhatemp[str(cid)+"titulo"], linhatemp[str(cid)+"data"], linhatemp[str(cid)+"detalhes"], 0, selecionarComando)
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        del linhatemp[str(cid)+"titulo"]
        del linhatemp[str(cid)+"data"]
        del linhatemp[str(cid)+"detalhes"]
        with open('linhatemp/{}.txt'.format(cid), 'w') as f:
            f.write("")
```

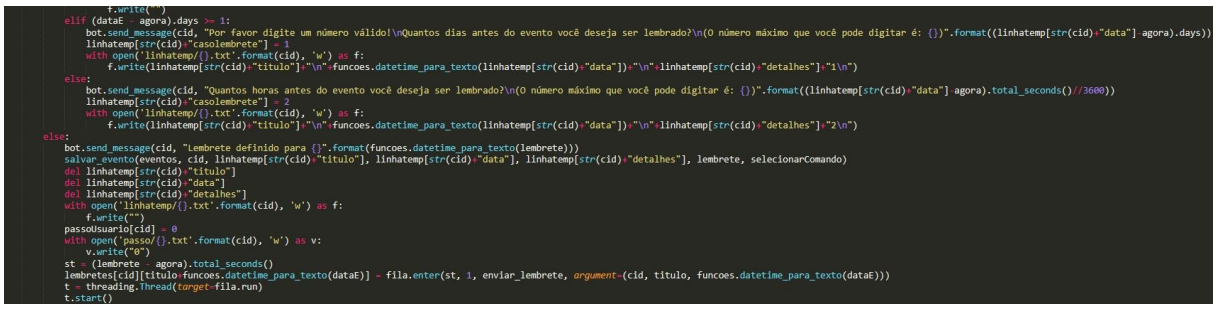
Figura 11.1- Função feita para o passo 6

Valida o tempo, exclui do arquivo ‘usuarios.txt’ as informações que não irão para o lembrete.

```
else:
    try:
        tempo = int(mensagem.text)
        if tempo < 0:
            raise ValueError
        dataE = linhatemp[str(cid)+"data"]
        titulo = linhatemp[str(cid)+"titulo"]
        if linhatemp[str(cid)+"casolembrete"] == 1:
            lembrete = dataE - datetime.timedelta(days=tempo)
        else:
            lembrete = dataE - datetime.timedelta(hours=tempo)
        agora = datetime.datetime.now()
        if lembrete < agora:
            if dataE < agora:
                bot.send_message(cid, "Não é mais possível criar um lembrete.")
                salvar_evento(eventos, cid, linhatemp[str(cid)+"titulo"], linhatemp[str(cid)+"data"], linhatemp[str(cid)+"detalhes"], 0, selecionarComando)
                passoUsuario[cid] = 0
                with open('passo/{}.txt'.format(cid), 'w') as v:
                    v.write("0")
                del linhatemp[str(cid)+"titulo"]
                del linhatemp[str(cid)+"data"]
                del linhatemp[str(cid)+"detalhes"]
                with open('linhatemp/{}.txt'.format(cid), 'w') as f:
                    f.write("")
```

Figura 11.2- Parte da função passo 6

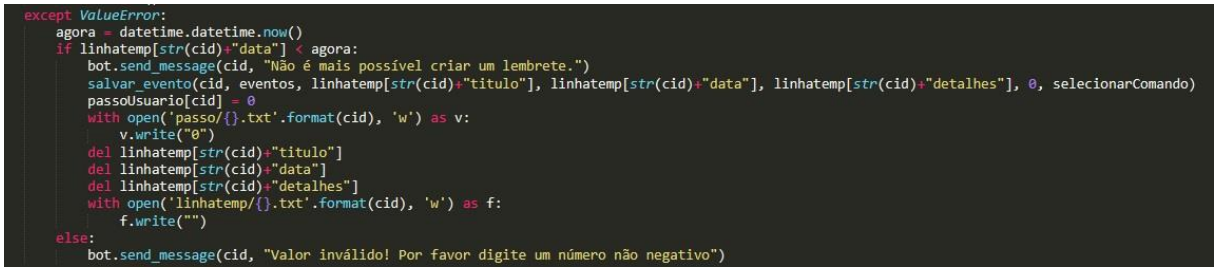
Ratifica a validade da data do lembrete, caso o usuário digite um número inválido, receberá uma mensagem para o usuário digitar um número válido.



```
            f.write("")
    elif (dataE - agora).days >= 1:
        bot.send_message(cid, "Por favor digite um número válido!\nQuantos dias antes do evento você deseja ser lembrado?\n(O número máximo que você pode digitar é: {})".format((linhatemp[str(cid)+"data"] - agora).days))
        linhatemp[str(cid)+"casolembrete"] = 1
        with open('linhatemp/{}.txt'.format(cid), 'w') as f:
            f.write(linhatemp[str(cid)+"titulo"]+"\n"+funcoes.datetime_para_texto(linhatemp[str(cid)+"data"])+"\n"+linhatemp[str(cid)+"detalhes"]+"1\n")
    else:
        bot.send_message(cid, "Quantos horas antes do evento você deseja ser lembrado?\n(O número máximo que você pode digitar é: {})".format((linhatemp[str(cid)+"data"] - agora).total_seconds()//3600))
        linhatemp[str(cid)+"casolembrete"] = 2
        with open('linhatemp/{}.txt'.format(cid), 'w') as f:
            f.write(linhatemp[str(cid)+"titulo"]+"\n"+funcoes.datetime_para_texto(linhatemp[str(cid)+"data"])+"\n"+linhatemp[str(cid)+"detalhes"]+"2\n")
else:
    bot.send_message(cid, "Lembrete definido para {}".format(funcoes.datetime_para_texto(lembrete)))
    salvar_evento(eventos, cid, linhatemp[str(cid)+"titulo"], linhatemp[str(cid)+"data"], linhatemp[str(cid)+"detalhes"], lembrete, selecionarComando)
    del linhatemp[str(cid)+"titulo"]
    del linhatemp[str(cid)+"data"]
    del linhatemp[str(cid)+"detalhes"]
    with open('linhatemp/{}.txt'.format(cid), 'w') as f:
        f.write("")
    passoUsuario[cid] = 0
    with open('passo/{}.txt'.format(cid), 'w') as v:
        v.write("0")
    st = (lembrete - agora).total_seconds()
    lembretes[cid][titulo+funcoes.datetime_para_texto(dataE)] = fila.enter(st, 1, enviar_lembrete, argument=(cid, titulo, funcoes.datetime_para_texto(dataE)))
    t = threading.Thread(target=fila.run)
    t.start()
```

Figura 11.3- Parte da função passo 6

Caso dê erro, irá validar novamente.



```
except ValueError:
    agora = datetime.datetime.now()
    if linhatemp[str(cid)+"data"] < agora:
        bot.send_message(cid, "Não é mais possível criar um lembrete.")
        salvar_evento(cid, eventos, linhatemp[str(cid)+"titulo"], linhatemp[str(cid)+"data"], linhatemp[str(cid)+"detalhes"], 0, selecionarComando)
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        del linhatemp[str(cid)+"titulo"]
        del linhatemp[str(cid)+"data"]
        del linhatemp[str(cid)+"detalhes"]
        with open('linhatemp/{}.txt'.format(cid), 'w') as f:
            f.write("")
    else:
        bot.send_message(cid, "Valor inválido! Por favor digite um número não negativo")
```

Figura 11.4- Parte da função passo 6

3.4 Lista de eventos

A função ‘lista_de_eventos’ serve para o bot enviar uma mensagem dizendo ‘não há eventos salvos’ caso não tenha e pergunta o que será feito em seguida ou mostrar uma lista de todos os eventos salvos e perguntar qual deles o usuário deseja editar.

```
@bot.message_handler(func=lambda message: message.text == comandos[1] and passo_do_usuario(message.chat.id) == 0)
def lista_de_eventos(mensagem):
    cid = mensagem.chat.id
    if len(eventos[cid]) == 0:
        bot.send_message(cid, "Não há eventos salvos", reply_markup=esconderTeclado)
        bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
    else:
        texto1 = "Selecione qual evento deseja editar:\n\n"
        for i in range(len(eventos[cid])):
            texto1 += "{0}) {1}: {2}\n".format(i+1, eventos[cid][i][0], funcoes.datetime_para_texto(eventos[cid][i][1]))
        bot.send_message(cid, texto1, reply_markup=teclado_editar(len(eventos[cid])))
        passoUsuario[cid] = 10
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("10")
```

Figura 1.1- Função para editar a lista de eventos

Figura 1.2- Imagem ilustrativa para mostrar que não há nada na lista de eventos

A função ‘teclado_editar’ é para quando o usuário selecionar a opção ‘lista de eventos’, então, aparecerá um teclado com a quantidade de eventos já criados:

```
def teclado_editar(quantidade_items):
    tecladoEditar = types.ReplyKeyboardMarkup()
    tecladoEditar.add(*[str(n)+")" for n in range(1, quantidade_items + 1)]+["CANCELAR"])
    return tecladoEditar
```

Figura 2.1- Função para criar um teclado com a quantidade de eventos mais um ícone de ‘cancelar’

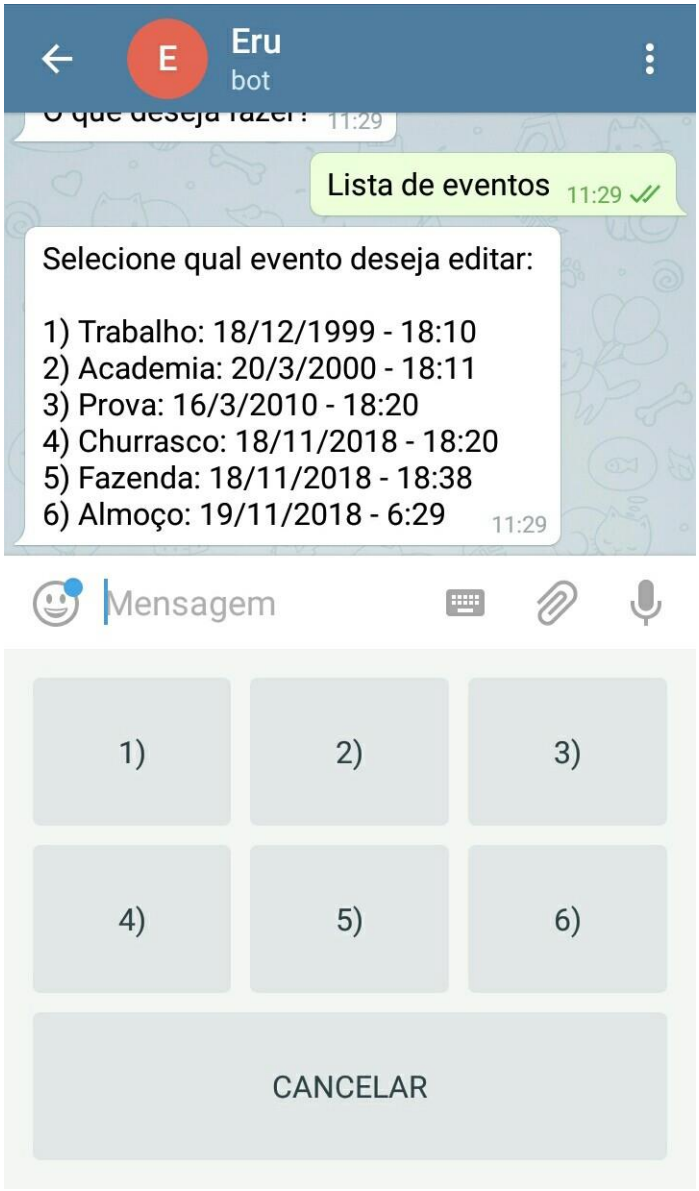


Figura 2.2- amostra de como fica o teclado com a quantidade de eventos criados.

A função ‘editar’ não exatamente edita, mas começa a edição, mostrará ao usuário uma lista de eventos anteriores e perguntará qual deles o usuário gostaria de editar.

```python
@bot.message_handler(func=lambda message: passo_do_usuario(message.chat.id) == 10)
def editar(mensagem):
    cid = mensagem.chat.id
    if mensagem.text in [str(n)+")" for n in range(1, len(eventos[cid]) + 1)]:
        edt = int(mensagem.text[:-1]) - 1
        editartemp[cid] = edt
        with open('editartemp/{}.txt'.format(cid), 'w') as f:
            f.write(str(edt))
        if eventos[cid][edt][3] == 0:
            texto2 = "Título:\n{}\n\nData:\n{}\n\nDetalhes:\n{}\n\nLembrete:\nNão definido".format(eventos[cid][edt][0],
                                        funcoes.datetime_para_texto(eventos[cid][edt][1]), eventos[cid][edt][2])
        else:
            texto2 = "Título:\n{}\n\nData:\n{}\n\nDetalhes:\n{}\n\nLembrete:\n{}".format(eventos[cid][edt][0],
         funcoes.datetime_para_texto(eventos[cid][edt][1]), eventos[cid][edt][2], funcoes.datetime_para_texto(eventos[cid][edt][3]))
        bot.send_message(cid, texto2, reply_markup=opcoesTeclado)
        passoUsuario[cid] = 12
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("12")
    elif mensagem.text == "CANCELAR":
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        bot.send_message(cid, "O que gostaria de fazer?", reply_markup=selecionarComando)
    else:
        bot.send_message(cid, "Por favor selecione uma das opções", reply_markup=teclado_editar(len(eventos[cid])))
```

Figura 3.1- Função de pré-edição dos eventos

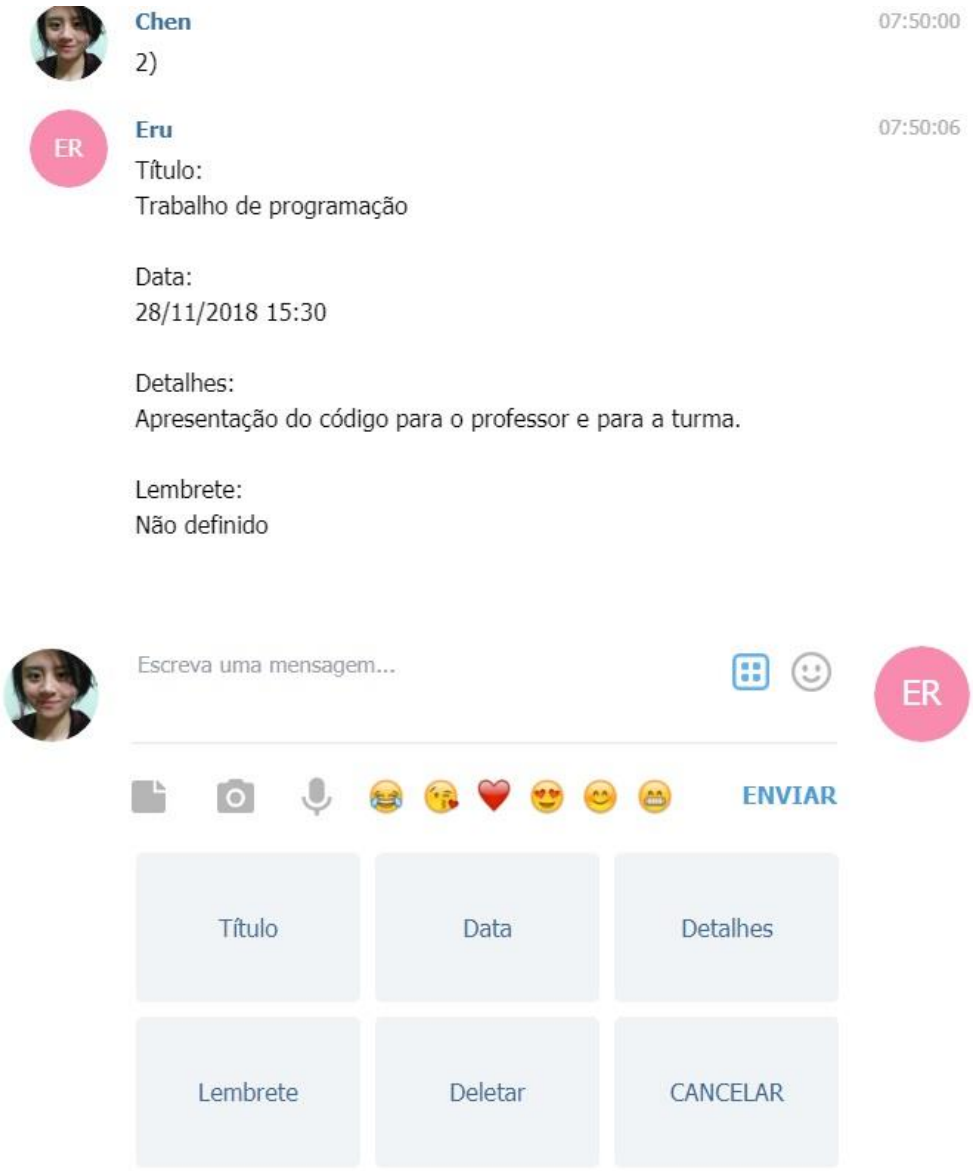


Figura 3.2- Amostra da imagem de qual opção o usuário deseja editar

A função ‘editar2’ do passo 12 é uma continuação da outra função ‘editar’, caso o usuário escolha ‘título’, então o bot mandará uma mensagem pedindo um novo título; caso seja ‘data’, a escolha, então primeiro, pedirá para inserir uma nova data;

```
@bot.message_handler(func=lambda message: passo_do_usuario(message.chat.id) == 12)
def editar2(mensagem):
    cid = mensagem.chat.id
    agora = datetime.datetime.now()
    if mensagem.text == "Título":
        bot.send_message(cid, "Digite o novo título:", reply_markup=esconderTeclado)
        editartempcaso[cid] = 1
        with open('editartempcaso/{}.txt'.format(cid), 'w') as f:
            f.write("1")
        passoUsuario[cid] = 13
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("13")
    elif mensagem.text == "Data":
        bot.send_message(cid, "Digite a nova data (no formato DD/MM/AAAA):", reply_markup=esconderTeclado)
        editartempcaso[cid] = 2
        with open('editartempcaso/{}.txt'.format(cid), 'w') as f:
            f.write("2")
        passoUsuario[cid] = 13
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("13")
```

Figura 4.1- parte da função para editar os eventos do passo 12

Caso a escolha de edição seja ‘detalhes’ então o usuário digitará novos detalhes; se o usuário escolher a opção ‘lembrete’, caso o evento seja antecedente ao tempo de agora, então não será possível editar um lembrete.

```
elif mensagem.text == "Detalhes":
    bot.send_message(cid, "Digite os novos detalhes:", reply_markup=esconderTeclado)
    editartempcaso[cid] = 3
    with open('editartempcaso/{}.txt'.format(cid), 'w') as f:
        f.write("3")
    passoUsuario[cid] = 13
    with open('passo/{}.txt'.format(cid), 'w') as v:
        v.write("13")
elif mensagem.text == "Lembrete":
    if (eventos[cid][editartemp[cid]][1] < datetime.datetime.now()):
        bot.send_message(cid, "Não é possível fazer isso! Esse evento já aconteceu!", reply_markup=esconderTeclado)
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        del editartemp[cid]
        with open('editartemp/{}.txt'.format(cid), 'w') as f:
            f.write("")
        bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
```

Figura 4.2- parte da função para editar os eventos do passo 12

Caso o evento inserido seja posterior à data de hoje, então o bot perguntará a antecedência para o lembrete em dias ou horas.

```
    bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
elif (eventos[cid][editartemp[cid]][1] - datetime.datetime.now()).days >= 1:
    bot.send_message(cid, "Quantos dias antes do evento você deseja ser lembrado?\n(O número máximo que você pode digitar é: {})"
        .format((eventos[cid][editartemp[cid]][1]-agora).days), reply_markup=esconderTeclado)
    editartempcaso[cid] = 4
    with open('editartempcaso/{}.txt'.format(cid), 'w') as f:
        f.write("4")
    passoUsuario[cid] = 13
    with open('passo/{}.txt'.format(cid), 'w') as v:
        v.write("13")
else:
    bot.send_message(cid, "Quantos horas antes do evento você deseja ser lembrado?\n(O número máximo que você pode digitar é: {})"
        .format((eventos[cid][editartemp[cid]][1]-agora).total_seconds()//3600), reply_markup=esconderTeclado)
    editartempcaso[cid] = 5
    with open('editartempcaso/{}.txt'.format(cid), 'w') as f:
        f.write("5")
    passoUsuario[cid] = 13
    with open('passo/{}.txt'.format(cid), 'w') as v:
        v.write("13")
```

Figura 4.3- parte da função para editar os eventos do passo 12

Caso a opção escolhida seja ‘deletar’ ou ‘cancelar’, o processo é mais rápido e óbvio, o evento será deletado ou cancelado.

```
            v.write("13")
elif mensagem.text == "Deletar":
    if eventos[cid][editartemp[cid]][3] != 0:
        if datetime.datetime.now() < eventos[cid][editartemp[cid]][3]:
            fila.cancel(lembretes[cid][eventos[cid][editartemp[cid]][0]+funcoes.datetime_para_texto(eventos[cid][editartemp[cid]][1])])
    del eventos[cid][editartemp[cid]]
    funcoes.salvar_edicao(eventos, cid)
    del editartemp[cid]
    with open('editartemp/{}.txt'.format(cid), 'w') as f:
        f.write("")
    passoUsuario[cid] = 0
    with open('passo/{}.txt'.format(cid), 'w') as v:
        v.write("0")
    bot.send_message(cid, "Evento deletado com sucesso!", reply_markup=esconderTeclado)
    bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
elif mensagem.text == "CANCELAR":
    passoUsuario[cid] = 0
    with open('passo/{}.txt'.format(cid), 'w') as v:
        v.write("0")
    bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
else:
    bot.send_message(cid, "Por favor selecione uma das opções", reply_markup=opcoesTeclado)
```

Figura 4.4- parte da função para editar os eventos do passo 12

A função ‘editar2’ do passo 13 serve para cancelar um lembrete, se tiver, e programa um novo e então continua perguntando o que o usuário quer fazer em sequência.

```
@bot.message_handler(func=lambda message: passo_do_usuario(message.chat.id) == 13)
def editar2(m):
    cid = m.chat.id
    if editartempcaso[cid] == 1:
        tituloNovo = m.text
        if eventos[cid][editartemp[cid]][3] != 0:
            if datetime.datetime.now() < eventos[cid][editartemp[cid]][3]:
                fila.cancel(lembretes[cid][eventos[cid][editartemp[cid]][0]+funcoes.datetime_para_texto(eventos[cid][editartemp[cid]][1])])
                st = (eventos[cid][editartemp[cid]][3] - datetime.datetime.now()).total_seconds()
                lembretes[cid][tituloNovo+funcoes.datetime_para_texto(eventos[cid][editartemp[cid]][1])] = fila.enter(st, 1, enviar_lembrete, argument=(cid,
                                                                                    tituloNovo, funcoes.datetime_para_texto(eventos[cid][editartemp[cid]][1])))
                t = threading.Thread(target=fila.run)
                t.start()
        eventos[cid][editartemp[cid]][0] = tituloNovo
        funcoes.salvar_edicao(eventos, cid)
        bot.send_message(cid, "Título editado com sucesso!")
        del editartempcaso[cid]
        with open('editartempcaso/{}.txt'.format(cid), 'w') as f:
            f.write("")
        del editartemp[cid]
        with open('editartemp/{}.txt'.format(cid), 'w') as f:
            f.write("")
        passoUsuario[cid] = 0
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("0")
        bot.send_message(cid, "O que deseja fazer?", reply_markup=selecionarComando)
```

Figura 5.1- parte da função editar2 do passo 13

Caso o usuário queira mudar a data, então a função irá validar a data, dizendo ao usuário se teve sucesso ou não. (A edição da data é composta de 3 fases)

Figura 5.2- parte da função editar2 no passo 13

Irá cancelar o detalhe antigo, programará um novo e então enviará uma mensagem de ‘Detalhes editados com sucesso’.

Figura 5.3- parte da função editar2 do passo 13

A figura seguinte irá validar os lembretes inválidos, como ‘não é possível criar um lembrete’ ou ‘ número inválido’...

Figura 5.4- parte da função editar2 do passo 13

Caso o lembrete possa realmente ser definido, então irá aparecer uma mensagem para o usuário para quando o lembrete foi definido e perguntará o que fazer em seguida, ou peça para o usuário digitar um número válido.

Figura 5.5- parte da função editar2 do passo 13

A função ‘editar_horario’ serve para a segunda fase da data, irá validar o horário.

Figura 6.1- parte da função editar_horario

Valida a data pergunta se quer um lembrete.

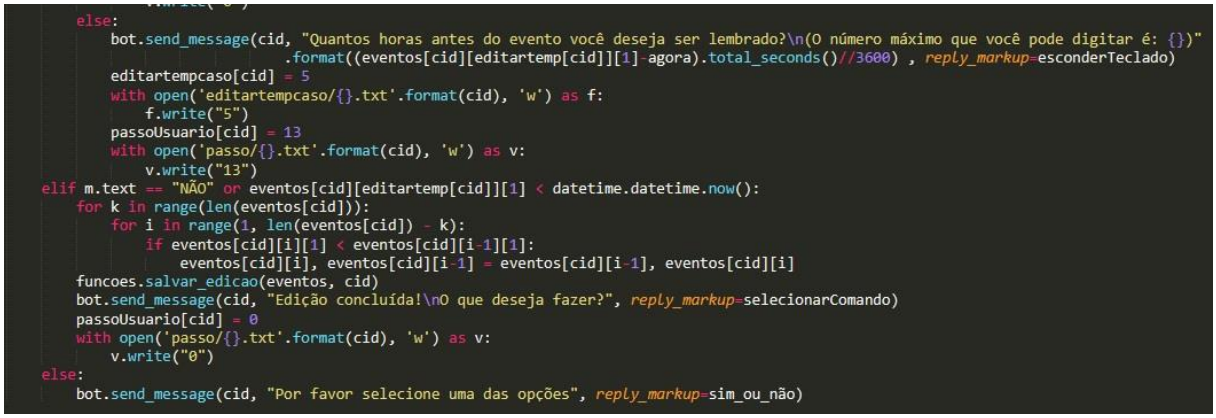
Figura 6.2- parte da função editar_horario

A função ‘editar_horario2’ consiste em criar o lembrete caso a resposta anterior seja ‘sim’.

```
@bot.message_handler(func=Lambda message: passo_do_usuario(message.chat.id) == 15)
def editar_horario2(m):
    cid = m.chat.id
    if m.text == "SIM":
        agora = datetime.datetime.now()
        if (eventos[cid][editartemp[cid]][1]-agora).days >= 1:
            bot.send_message(cid, "Quantos dias antes do evento você deseja ser lembrado?\n(O número máximo que você pode digitar é: {})"
                                                    .format((eventos[cid][editartemp[cid]][1]-agora).days), reply_markup=esconderTeclado)
            editartempcaso[cid] = 4
            with open('editartempcaso/{}.txt'.format(cid), 'w') as f:
                f.write("4")
            passoUsuario[cid] = 13
            with open('passo/{}.txt'.format(cid), 'w') as v:
                v.write("13")
        elif eventos[cid][editartemp[cid]][1] < agora:
            bot.send_message(cid, "Não é mais possível criar um lembrete.")
            for k in range(len(eventos[cid])):
                for i in range(1, len(eventos[cid]) - k):
                    if eventos[cid][i][1] < eventos[cid][i-1][1]:
                        eventos[cid][i], eventos[cid][i-1] = eventos[cid][i-1], eventos[cid][i]
            funcoes.salvar_edicao(eventos, cid)
            bot.send_message(cid, "Edição concluída!\nO que deseja fazer?", reply_markup=selecionarComando)
            passoUsuario[cid] = 0
            with open('passo/{}.txt'.format(cid), 'w') as v:
                v.write("0")
```

Figura 7.1- parte da função editar_horario caso a resposta seja ‘sim’

Caso a resposta anterior seja ‘não’, então encerra a edição e caso o usuário não responda nem sim e nem não, então o programa pedirá ao usuário escolher uma das opções.



```
    else:
        bot.send_message(cid, "Quantos horas antes do evento você deseja ser lembrado?\n(O número máximo que você pode digitar é: {})"
                                .format((eventos[cid][editartemp[cid]][1]-agora).total_seconds()//3600) , reply_markup=esconderTeclado)
        editartempcaso[cid] = 5
        with open('editartempcaso/{}.txt'.format(cid), 'w') as f:
            f.write("5")
        passoUsuario[cid] = 13
        with open('passo/{}.txt'.format(cid), 'w') as v:
            v.write("13")
elif m.text == "NÃO" or eventos[cid][editartemp[cid]][1] < datetime.datetime.now():
    for k in range(len(eventos[cid])):
        for i in range(1, len(eventos[cid]) - k):
            if eventos[cid][i][1] < eventos[cid][i-1][1]:
                eventos[cid][i], eventos[cid][i-1] = eventos[cid][i-1], eventos[cid][i]
    funcoes.salvar_edicao(eventos, cid)
    bot.send_message(cid, "Edição concluída!\nO que deseja fazer?", reply_markup=selecionarComando)
    passoUsuario[cid] = 0
    with open('passo/{}.txt'.format(cid), 'w') as v:
        v.write("0")
else:
    bot.send_message(cid, "Por favor selecione uma das opções", reply_markup=sim_ou_não)
```

Figura 7.2- parte da função editar_horario caso a resposta não seja ‘sim’

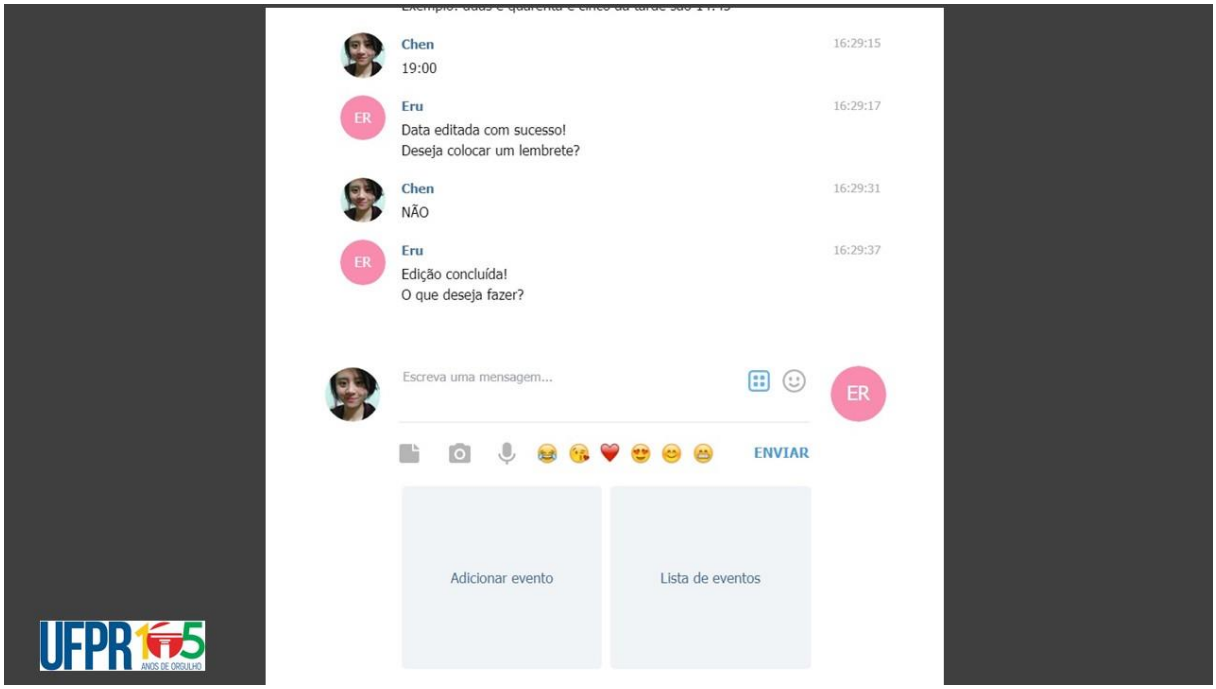


Figura 7.2.1- Amostra do que ocorre caso a edição é concluída

4 CONCLUSÃO

O trabalho tinha o intuito de criar algo usando o Bot do Telegram, e então a gente decidiu criar uma agenda que executaria no Bot, assim, poderíamos procurar funções novas, por exemplo, o ‘sched’ ou então, conhecer o aplicativo Telegram. No fim, conseguimos criar um código que funciona perfeitamente. Se o código já não estivesse tão grande, talvez pudéssemos ter implementado mais coisas, mas por enquanto é isso.

REFERÊNCIAS

PYTHON. Docs. **datetime — Basic date and time types.** Curitiba, 2018 em: https://docs.python.org/3/library/datetime.html. Acesso: 26 nov. 2018

PYTHON. Docs. **sched — Event scheduler.** Curitiba, 2018 em : https://docs.python.org/3/library/sched.html. Acesso: 26 nov. 2018

PYTHON. PyMOTW. **sched – Generic event scheduler**. Curitiba, 2018 em: https://pymotw.com/2/sched/.Acesso: 26 nov. 2018

UNIVERSIDADE FEDERAL DO PARANA (UFPR). Ava-moodle. **Exemplo – Relatório Programação.** Curitiba, 2018 em: https://ava.ufpr.br/pluginfile.php/195413/mod_resource/content/0/Exemplo%20-%20Relat%C3%B3rio%20Programa%C3%A7%C3%A3o.pdf. Acesso: 17 nov. 2018.

UNIVERSIDADE FEDERAL DO PARANA (UFPR). Sistema de Bibliotecas. **Orientação para Normalização de Trabalhos Acadêmicos**. Curitiba, 2018 em: https://www.portal.ufpr.br/normalizacao. Acesso: 17 nov. 2018.

UNIVERSIDADE FEDERAL DO PARANA (UFPR). Portal UFPR. **Referências – exemplos**. Curitiba, 2018 em : < https://www.portal.ufpr.br/tutoriais/tutoriais_normaliza/referencia_exemplo.pdf>. Acesso: 17 nov. 2018.

WIKIPEDIA. Site. **TELEGRAM (APLICATIVO)**. Curitiba, 2018 em : https://pt.wikipedia.org/wiki/Telegram_(aplicativo).Acesso: 26 nov. 2018.