Faculdade de Informática e Administração Paulista

DISRUPTIVE

ARCHITECTURES: IOT, IOB & IA

SPRINT 2

Declaração das variáveis:

r = reconhecedor do speech recognition
agora = chamar a função do datetime
data = formatar para receber apenas dia/mês/ano
hora = formatar para receber apenas hora:minuto

```
#Inicializar o reconhecedor!
r = sr.Recognizer()
agora = datetime.now()
data = agora.strftime("%d/%m/%Y")
hora = agora.strftime("%H:%M")
```

Função "*sf fala*" é criada e usando o *speech-to-text* do Google, salvando um arquivo com um áudio e após executá-lo o arquivo é apagado logo em seguida

```
def st_fala(audio_string):
    tts = gTTS(text = audio_string, lang ='pt')
    r = random.randint(1,100000000)
    arq_audio = "audio-"+ str(r) + ".mp3"
    tts.save (arq_audio)
    playsound.playsound (arq_audio)
    print(audio_string)
    os.remove(arq_audio)
```

FIAP

Função "*gravar_áudio*" é criada para detectar o microfone e utilizá-lo como source e para detectar a voz do usuário.

```
def gravar_audio(rqr = False):
    with sr.Microphone() as source: #Usando o microfone como source
        if rqr:
            st_fala(rqr)
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source, phrase_time_limit = 5)
        voice_data = ""
        try:
            voice_data = r.recognize_google(audio,language='pt-br')
            print(voice_data)
        except sr.UnknownValueError:
            st_fala("Não compreendi")
        except sr.RequestError:
            st_fala("Infelizmente vivemos no Brasil e o serviço caiu")
        return voice_data
```

Função criada para criar as instruções da assistente, como: qual é o seu nome,pesquisar e etc.

```
def respostas(voice_data):
```

Instrução nome retorna a função de fala com a String “Me chamo Strix!”

```
    if "nome" in voice_data:
        st_fala("Me chamo Strix!")
```

Instrução data retorna a função de fala com as Strings “Hoje é dia: “ e “e o horário é:” concatenadas com as variáveis data e hora.

```
    if "data" in voice_data:
        st_fala("hoje é dia: " + data + "e o horário é:" + hora)
```

Instrução assistir retorna guarda em uma variável chamada pesquisa o resultado da função gravar_audio que irá gravar o que a pessoa falar, logo em seguida irá armazenar na variável url o link da query de pesquisa do youtube e abrir no navegador.

```
if "assistir" in voice_data:
    pesquisa = gravar_audio("O que você quer assistir?")
    url = "https://youtube.com/results?search_query=" + pesquisa
    webbrowser.get().open(url)
    st_fala("Isso foi o que encontrei para " + pesquisa)
```

Instrução assistir retorna guarda em uma variável chamada pesquisa o resultado da função gravar_audio que irá gravar o que a pessoa falar, logo em seguida irá armazenar na variável url o link da query de pesquisa no google e abrir no navegador.

```
if "pesquisar" in voice_data:
    pesquisa = gravar_audio("O que você quer pesquisar?")
    url = "https://google.com/search?q=" + pesquisa
    webbrowser.get().open(url)
    st_fala("Isso foi o que encontrei para "+ pesquisa)
```

Instrução anime receberá uma variável anime onde será armazenado o que o usuário deseja pesquisa e usando a *mal-api* irá pesquisa no site *MyAnimeList* o nome do anime e retornar o título junto com a nota e também irá abrir o navegador do usuário com a página de pesquisa do *MyAnimeList.*

```
    if "anime" in voice_data:
        anime = gravar_audio("Qual anime deseja pesquisar?")
        search = AnimeSearch(anime)
        st_fala(search.results[0].title)
        url = "https://myanimelist.net/search/all?q=" + anime +"&cat=all"
        webbrowser.get().open(url)
        st_fala("A nota do anime é: " + str(search.results[0].score))
```

Instrução sair irá retornar a fala “Saindo…” e caso a tela da câmera de detecção facial ainda esteja aberta, encerrará o processo e logo em seguida terminará a execução do programa

```
    if "sair" in voice_data:
        st_fala("Saindo...")
        camera.release()
        cv2.destroyAllWindows()
        exit()
```

É criado um laço de repetição para detectar e reconhecer um rosto e caso tenha um rosto reconhecido abrirá as demais funcionalidades da aplicação.

```
camera = cv2.VideoCapture(0)
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')



while 1:
    ret, img = camera.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for(x, y, w, h) in faces:
        cv2.rectangle(img, (x,y), (x +w, y+h), (255,0,0), 2)
        roi_gray = gray[y:y + h, x:x + w]
        roi_color = img[y:y +h, x:x + w]


        camera.release()
        cv2.destroyAllWindows()
        st_fala("Rosto detectado! Olá, como posso ajudar?")
        time.sleep(1)
        while 1:
            voice_data = gravar_audio()
            respostas(voice_data)


    cv2.imshow('img', img)
    k = cv2.waitKey(1) & 0xff
    if k == 27:
        break
```