Assistente Virtual por Voz

QuantumFinance

Trabalho em Python — Atendimento Digital com Inteligência de Voz

Disciplina: Audio Recognition Workshop.
Prof. Alexandre Gastaldi Lopes Fernandes
Turma: 10 DTS – MBA -Data Science and Artificial Intelligence

Grupo: Lucas Santos RM 358024
Rafael Gallo RM 358285
Társis Fortes Tavares RM358921

Introdução

Sistema de atendimento por voz

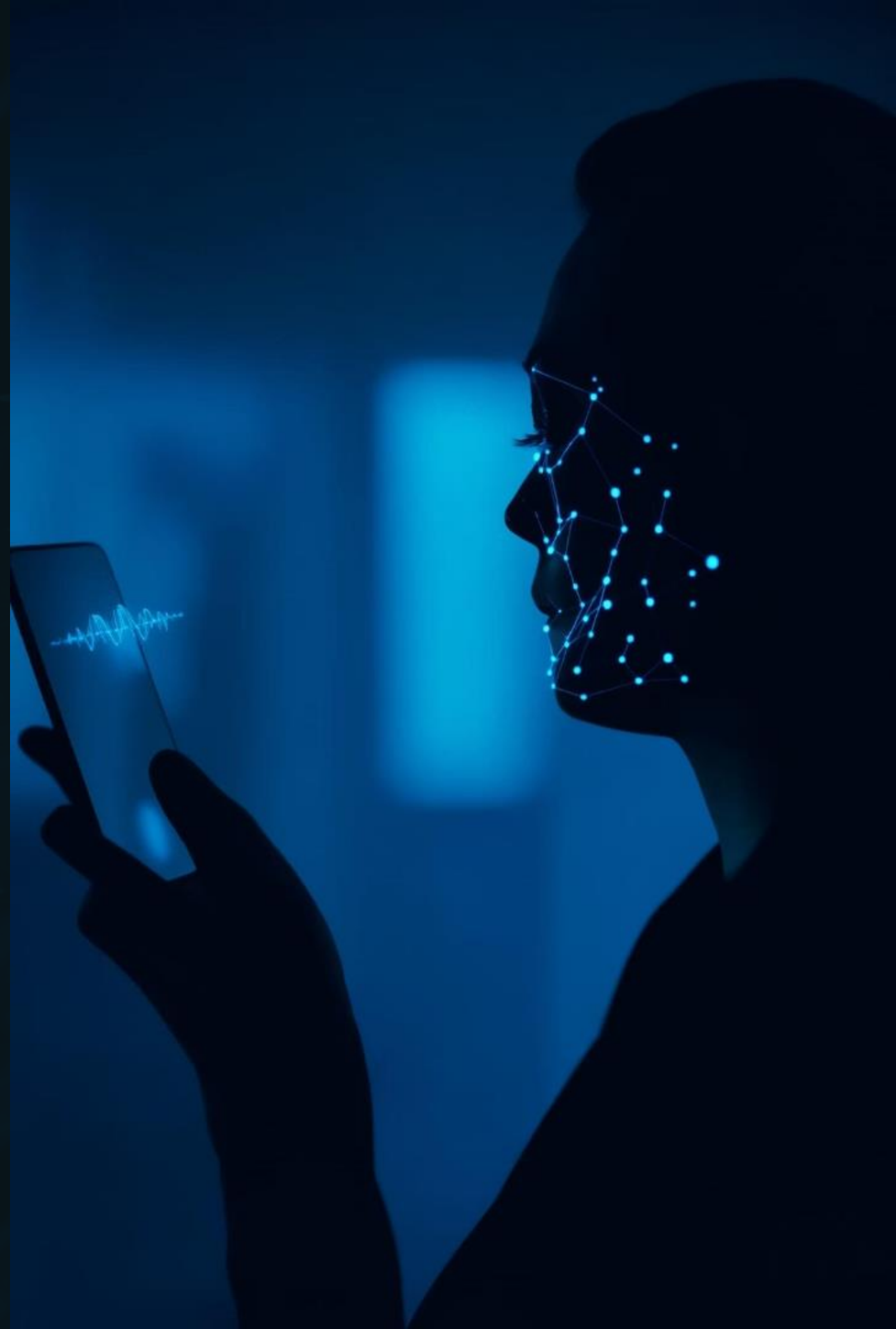
Simula call center da QuantumFinance

Funcionalidades

Saúda, oferece menu, interpreta voz, responde

Interação por voz

Cliente fala em vez de digitar



Bibliotecas Usadas

gTTS (Google Text-to Speech)

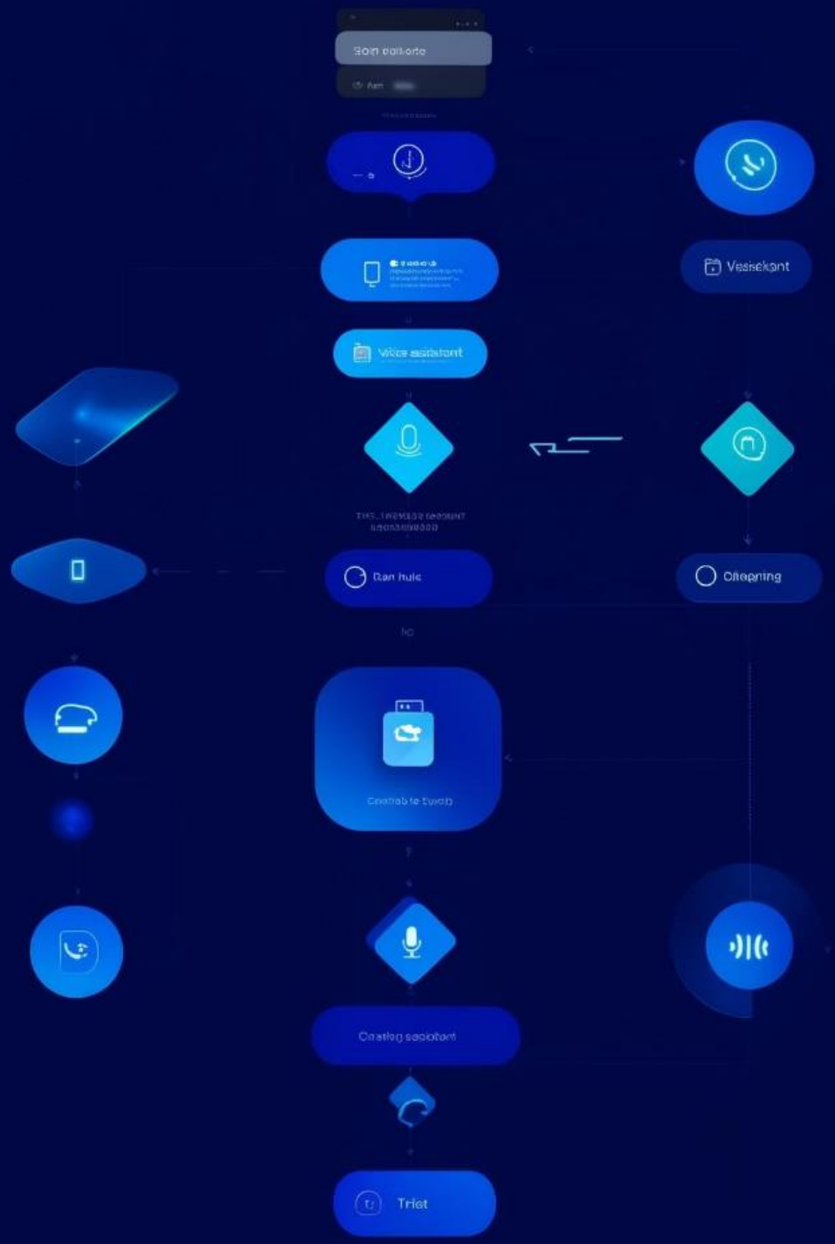
Converte texto em voz

speech_recognition (Speech To Text)

Converte fala em texto

pyaudio/playsound

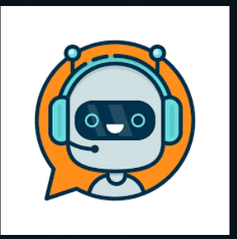
Capta e executa áudio



Funcionamento Geral

Mensagem inicial

Sistema toca opções



Cliente fala

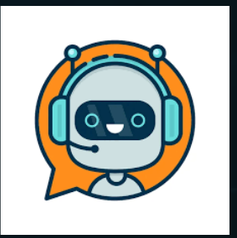
Escolhe opção por voz

Sistema interpreta

Entende comando falado

Confirmação

Toca áudio correspondente



Estrutura do Código

Importação

Bibliotecas necessárias

Funções de Suporte

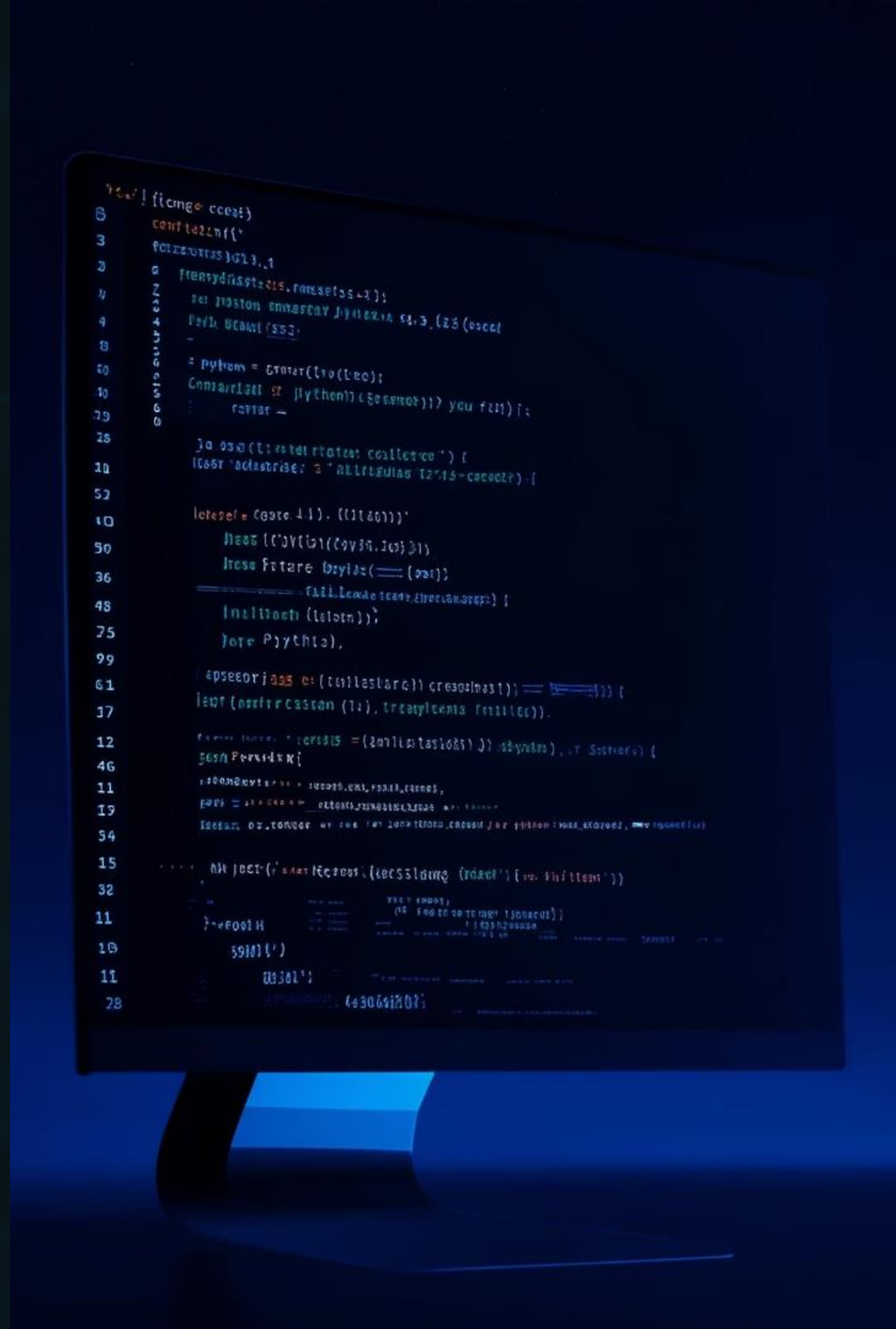
Ouvir microfone, Gerar áudio, tocar áudio.

Configuração

Geração dos arquivos de voz

Loop de Atendimento

Interações com cliente



Bloco 1 — Importação das bibliotecas



Importamos gTTS, speech_recognition, playsound, Os e time.

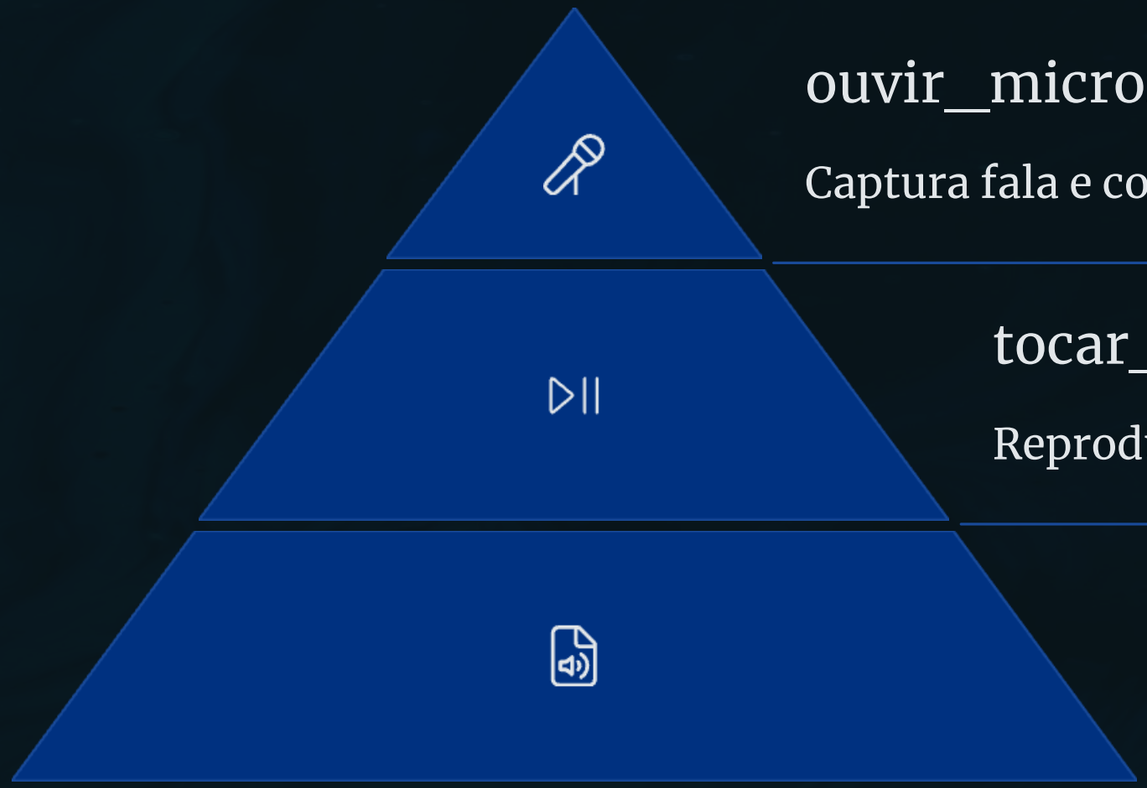
```python
# ================================================================================================================
#                                  Bloco 1 - Importação Bibliotecas
# ================================================================================================================
# Instalação das bibliotecas (foram executadas no terminal)
# pip install gTTs speechrecognition pyaudio playsound
# pip install pipwin
# pipwin install pyaudio
# pip install gTTs speechrecognition pyaudio playsound
# pip install playsound
# pip install playsound==1.2.2

#´Execução das bibliotecas instaladas
from gtts import gTTS
import os
import speech_recognition as sr
from playsound import playsound
import pygame #inserida depois que foi inserido o sample para teste
import time

#
```

Obs: A execução das bibliotecas pip install foram feitas diretamentes no prompt de comando.

Prompt de Comando

```
C:\Users\Samsung>pip install playsound==1.2.2
Collecting playsound==1.2.2
  Downloading playsound-1.2.2-py2.py3-none-any.whl.metadata (3.3 kB)
Downloading playsound-1.2.2-py2.py3-none-any.whl (6.0 kB)
Installing collected packages: playsound
Successfully installed playsound-1.2.2
```

Bloco 2 — Funções de Suporte



ouvir_microfone()

Captura fala e converte em texto

tocar_audio()

Reproduz arquivos de áudio

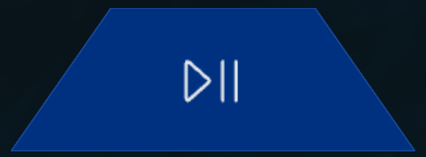
gerar_audio()

Transforma texto em áudio .mp3

```python
# ================================================================
#                                  Bloco 2 - Funções de suporte
# ================================================================
def ouvir_microfone():
    """Captura áudio do microfone e converte para texto."""
    recognizer = sr.Recognizer()

    with sr.Microphone() as source:
        print("Ouvindo...")
        audio = recognizer.listen(source, phrase_time_limit=10)
        print("Processando...")

    try:
        frase = recognizer.recognize_google(audio, language='pt-BR')
        print(f"Você disse: {frase}")
        return frase.lower()
    except sr.UnknownValueError:
        print("Não entendi o que foi dito.")
        return None
    except sr.RequestError:
        print("Erro no serviço de reconhecimento.")
        return None

def gerar_audio(texto, filename):
    """Gera um arquivo de áudio a partir de um texto."""
    tts = gTTS(text=texto, lang='pt-br')
    tts.save(filename)

def tocar_audio(filename):
    """Reproduz um arquivo de áudio."""
    playsound(filename)
```

ouvir_microfone()

Captura fala e converte em texto



tocar_audio()

Reproduz arquivos de áudio

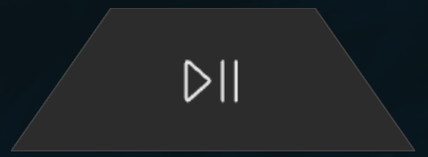
gerar_audio()

Transforma texto em áudio .mp3

```python
# ================================================================
#                          Bloco 2 - Funções de suporte
# ================================================================
def ouvir_microfone():
    """Captura áudio do microfone e converte para texto."""
    recognizer = sr.Recognizer()

    with sr.Microphone() as source:
        print("Ouvindo...")
        audio = recognizer.listen(source, phrase_time_limit=10)
        print("Processando...")

    try:
        frase = recognizer.recognize_google(audio, language='pt-BR')
        print(f"Você disse: {frase}")
        return frase.lower()
    except sr.UnknownValueError:
        print("Não entendi o que foi dito.")
        return None
    except sr.RequestError:
        print("Erro no serviço de reconhecimento.")
        return None

def gerar_audio(texto, filename):
    """Gera um arquivo de áudio a partir de um texto."""
    tts = gTTS(text=texto, lang='pt-br')
    tts.save(filename)

def tocar_audio(filename):
    """Reproduz um arquivo de áudio."""
    playsound(filename)
```

ouvir_microfone()

Captura fala e converte em texto



tocar_audio()

Reproduz arquivos de áudio

gerar_audio()

Transforma texto em áudio .mp3

```python
def ouvir_microfone():
    """Captura áudio do microfone e converte para texto."""
    recognizer = sr.Recognizer()

    with sr.Microphone() as source:
        print("Ouvindo...")
        audio = recognizer.listen(source, phrase_time_limit=10)
        print("Processando...")

    try:
        frase = recognizer.recognize_google(audio, language='pt-BR')
        print(f"Você disse: {frase}")
        return frase.lower()
    except sr.UnknownValueError:
        print("Não entendi o que foi dito.")
        return None
    except sr.RequestError:
        print("Erro no serviço de reconhecimento.")
        return None
```

Criação de um objeto usando a biblioteca speech_recognition

Ativa o microfone.
Exibe no console a mensagem “Ouvindo”
Começa a gravar a voz do usuário durante 10 segundos;
Se o usuário terminar de falar antes ele já processa;
Exibe no console “Processando”.

Inicia o bloco de tentativa. Onde utiliza o método recognize_google(), o qual envia o áudio capturado para os servidores da Google. O resultado é gravado na variável “frase”. O parâmetro ‘pt-BR’ define que queremos a transcrição em português do Brasil. Exibe no console o resultado. Transforma tudo para minúsculo.

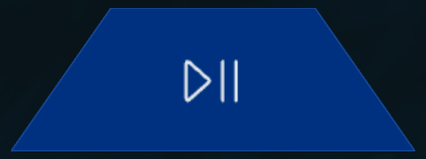
Esse erro acontece quando o áudio foi capturado, mas não foi possível entender nenhuma palavra (por ruído, fala confusa, microfone ruim, etc.

Esse erro acontece se não houver erro com a comunicação da API. Podendo ser falta de internet ou falha com o servidor.
Em caso de falha e “return None” a assistente irá refazer o loop das perguntas, conforme será explicado mais a frente.

```python
# ================================================================
#                          Bloco 2 - Funções de suporte
# ================================================================
def ouvir_microfone():
    """Captura áudio do microfone e converte para texto."""
    recognizer = sr.Recognizer()

    with sr.Microphone() as source:
        print("Ouvindo...")
        audio = recognizer.listen(source, phrase_time_limit=10)
        print("Processando...")

    try:
        frase = recognizer.recognize_google(audio, language='pt-BR')
        print(f"Você disse: {frase}")
        return frase.lower()
    except sr.UnknownValueError:
        print("Não entendi o que foi dito.")
        return None
    except sr.RequestError:
        print("Erro no serviço de reconhecimento.")
        return None

def gerar_audio(texto, filename):
    """Gera um arquivo de áudio a partir de um texto."""
    tts = gTTS(text=texto, lang='pt-br')
    tts.save(filename)

def tocar_audio(filename):
    """Reproduz um arquivo de áudio."""
    playsound(filename)
```

ouvir_microfone()

Captura fala e converte em texto

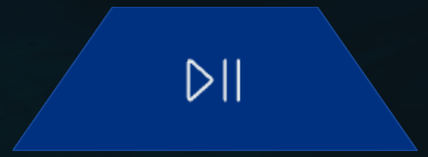


tocar_audio()

Reproduz arquivos de áudio

gerar_audio()

Transforma texto em áudio .mp3



tocar_audio()

Reproduz arquivos de áudio

```python
def gerar_audio(texto, filename):
    """Gera um arquivo de áudio a partir de um texto."""
    tts = gTTS(text=texto, lang='pt-br')
    tts.save(filename)
```

Definição de uma função para gerar áudio.
Onde o texto será convertido em áudio, em voz pt-br.
Após ser realizado a conversão ele será salvo como .mp3 armazenado pela variável filename que serão os nomes dos áudios criados para cada frase.

gerar_audio()

Transforma texto em áudio .mp3

```python
def tocar_audio(filename):
    """Reproduz um arquivo de áudio."""
    playsound(filename)
```

Definição de uma função para executar o áudio em .mp3 existente

Bloco 3 — Configuração dos Áudios

Saudação

Boas-vindas ao cliente

Menu de opções

Apresenta escolhas disponíveis

Confirmações

Resposta para cada opção

Erro e encerramento

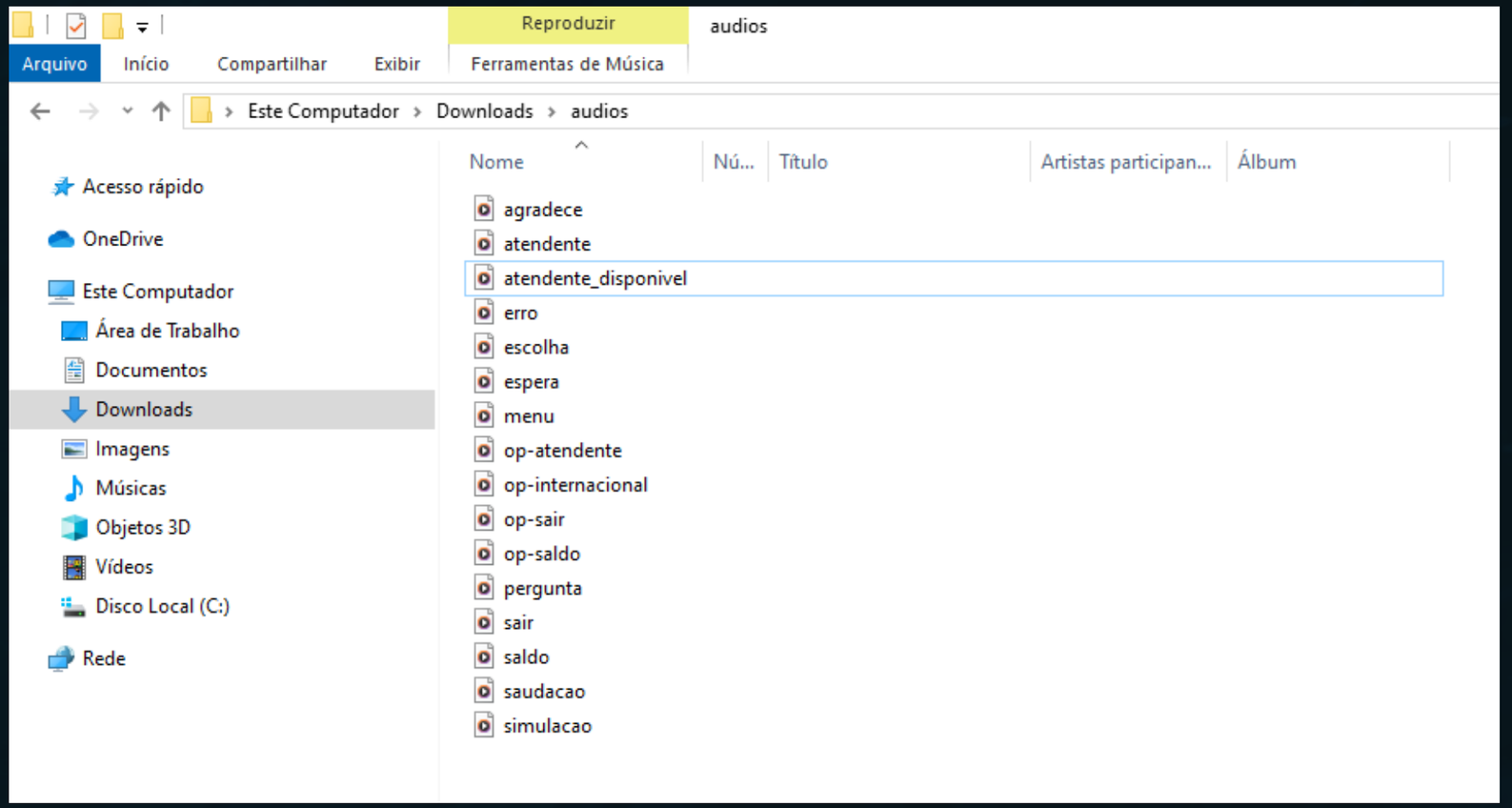
Mensagens de falha e despedida

```
# ================================================================================
#                                Bloco 3 - Configuração de áudios
# ================================================================================

# Pasta para armazenar os áudios
if not os.path.exists('audios'):
    os.makedirs('audios')

# Áudios principais
audios_textos = {
    "menu": "Bem-vindo! Meu nome é Samanta, assistente virtual da QuantumFinance. Por favor, diga uma das opções: "
            "Consulta ao saldo da conta, Simulação de compra internacional, "
            "Falar com um atendente ou Sair do atendimento.",
    "saldo": "Você escolheu Consulta ao saldo da conta. Seu saldo é de R$ 6.950,00",
    "simulacao": "Você escolheu Simulação de compra internacional. Certo, irei enviar a s
    "atendente": "Você escolheu Falar com um atendente. Aguarde na linha, pois os nossos                                    der.",
    "sair": "Encerrando o atendimento. A QuantumFinance agradece seu contato.",
    "erro": "Desculpe, não entendi. Por favor, repita sua opção.",
}

# Gerar os arquivos de áudio se ainda não existirem
for nome, texto in audios_textos.items():
    caminho = f'audios/{nome}.mp3'
    if not os.path.isfile(caminho):
        gerar_audio(texto, caminho)
```

Essa função verifica se há uma pasta existente com o nome áudios, ou não. Caso sim ela retorna True e se a pasta já existe False.
Utilizando a biblioteca os caso não existam áudios o algoritmo makedirs cria a pasta 'áudios'.

Criação de um dicionário com os nomes dos arquivos de áudio para cada ação do loop.

Criação de um laço for para gerar os arquivos de áudio (.mp3) dentro do diretório 'áudio'. Caso ele não existir.



Ele gerou os arquivos na pasta Downloads, pois é onde eu havia salvo o diretório do projeto no VSCode.

Se acaso eu quiser alterar alguma fala da assistente, é necessário que eu exclua o arquivo que quero alterar, para que ela possa gerar o áudio atualizado.

```
import os
print(os.getcwd())
```

Através dessa função pode-se checar onde os arquivos estão sendo salvos. No exemplo em questão:

```
PS C:\Users\Samsung\Downloads>
```

Bloco 4 — Loop de Atendimento

1

Tocar menu

Apresenta opções

Escutar usuário

Captura comando falado

2

Verificar palavras-chave

Identifica intenção

3

Responder

Toca áudio correspondente

4

```python
# ============================================================================================================
#                                        Bloco 4 - Loop de atendimento
# ============================================================================================================

def atendimento():
    while True:
        # Tocar menu de opções
        tocar_audio('audios/menu.mp3')

        # Ouvir usuário
        resposta = ouvir_microfone()

        if resposta is None:
            tocar_audio('audios/erro.mp3')
            continue

        # Verificar palavras-chave
        if any(palavra in resposta for palavra in ['saldo','conta','consulta','quanto tenho', 'quanto que eu tenho']):
            tocar_audio('audios/saldo.mp3')

        elif "simulação" in resposta or "compra internacional" in resposta or "simulacao" in resposta:
            tocar_audio('audios/simulacao.mp3')

        elif "atendente" in resposta or "falar com atendente" in resposta:
            tocar_audio('audios/atendente.mp3')
            print("Você está na fila de espera. Pressione CTRL+C para sair.")

            try:
                while True:
                    tocar_audio('audios/espera.mp3')  # toca o sample de espera
            except KeyboardInterrupt:
                    print("Atendente disponível. Saindo da espera...")
                    tocar_audio('audios/atendente_disponivel.mp3')
                    exit()

        elif "sair" in resposta or "encerrar" in resposta or "fechar" in resposta:
            tocar_audio('audios/sair.mp3')
            break
        else:
            tocar_audio('audios/erro.mp3')
if __name__ == "__main__":
    print("Iniciando atendimento...")
    atendimento()
```

Roda a função tocar_áudio com o dicionário criado menu com as opções.

```
"menu": "Bem-vindo! Meu nome é Samanta, assistente virtual da QuantumFinance. Por favor, diga uma das opções: "
        "Consulta ao saldo da conta, Simulação de compra internacional, "
        "Falar com um atendente ou Sair do atendimento.",
```

Roda a função com a ativação do microfone.

Se o valor de captação do áudio for nula, roda a frase de erro e executa o menu novamente

```
"erro": "Desculpe, não entendi. Por favor, repita sua opção.",
```

Verificação das palavras chaves para a opção de verificar saldo na conta; simulação internacional e falar com atendente

Quando selecionada a resposta de falar com atendente, cai na música de espera “sample”. Se selecionado o comando Ctrl+C interrompe o ciclo, simulando atendente_disponivel.

Bora ver na prática?

Link do vídeo:

https://www.youtube.com/watch?v=ckJCKkdiiJI

Figura gerada por IA.

Assistente Virtual da Quantum Finance Samantha

Encerramento

Demonstração prática

Sistema de atendimento com reconhecimento de voz

Tecnologias aplicadas

Text-to-Speech
e Speech-to-Text

Aprendizado

Aplicação real de Python para soluções de atendimento vitual