SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL
CENTRO DE FORMAÇÃO PROFISSIONAL FIDÉLIS REIS
CURSO DE DESENVOLVIMENTO DE SISTEMAS

TESTE DE SISTEMAS

RODRIGO BORGES

EXERCÍCIOS DE TESTE DE SISTEMAS

UBERABA - MG
2023

SUMÁRIO

1. INTRODUÇÃO

A música é uma parte fundamental da vida de muitas pessoas, e a tecnologia moderna trouxe uma revolução na forma como consumimos e compartilhamos essa forma de arte. Um aplicativo de música é uma ferramenta poderosa que coloca a vasta coleção de músicas do mundo ao alcance de nossas mãos, permitindo que os usuários descubram, ouçam e compartilhem suas músicas favoritas de maneira conveniente.

Nesta era digital, aplicativos de música desempenham um papel crucial, servindo como uma espécie de trilha sonora para nossas vidas, seja durante uma corrida matinal, uma viagem de carro, uma festa com amigos ou um momento de introspecção.

Assim, vamos tentar explorar as características, funcionalidades e impacto de um aplicativo de música. Desde a criação de playlists personalizadas que se adaptam ao nosso humor até a descoberta de novas faixas e artistas por meio de algoritmos de recomendação inteligentes, esses aplicativos têm a capacidade de moldar nossa experiência musical de maneira única.

2. APLICATIVO ORIGINAL SEM ALTERAÇÕES (com BUGS)

```
Python:import pygame
import tkinter as tk
from tkinter import filedialog
from tkinter import Listbox, Scrollbar, ttk

# Inicializa o Pygame.
pygame.init()

# Cria uma janela.
janela = tk.Tk()
janela.title("Player de Música")

# Cria uma lista para armazenar as músicas.
lista_de_reproducao = []
```

```
# Função para carregar uma música.
def carregar_musica():
        caminho_do_arquivo = filedialog.askopenfilename(title="Selecione uma música", filetypes=[("Arquivos de áudio", "*.mp3")])
    if caminho_do_arquivo:
        pygame.mixer.music.load(caminho_do_arquivo)
        lista_de_reproducao.append(caminho_do_arquivo)
        lista_musicas.insert(tk.END, caminho_do_arquivo)

# Função para tocar a música selecionada na lista.
def tocar_musica():
    indice_selecionado = lista_musicas.curselection()
    if indice_selecionado:
        musica_selecionada = lista_de_reproducao[indice_selecionado[0]]
        pygame.mixer.music.load(musica_selecionada)
        pygame.mixer.music.play()

# Função para interromper a música atual.
def parar_musica():
    pygame.mixer.music.stop()
    pygame.mixer.music.stop()  # Erro: Chamada dupla da função stop
    # Comentário: A função stop está sendo chamada duas vezes consecutivas

# Função para avançar para a próxima música.
def proxima_musica():
    indice_atual = lista_musicas.curselection()
    if indice_atual:
        indice_proxima = indice_atual[0] + 1
        if indice_proxima < len(lista_de_reproducao):
            musica_selecionada = lista_de_reproducao[indice_proxima]
            pygame.mixer.music.load(musica_selecionada)
            pygame.mixer.music.play()
```

```
            lista_musicas.select_clear(0, tk.END)
            lista_musicas.selection_set(indice_proxima)
            lista_musicas.activate(indice_proxima)

# Função para voltar para a música anterior.
def musica_anterior():
    indice_atual = lista_musicas.curselection()
    if indice_atual:
        indice_anterior = indice_atual[0] - 1
        if indice_anterior >= 0:
            musica_selecionada = lista_de_reproducao[indice_anterior]
            pygame.mixer.music.load(musica_selecionada)
            pygame.mixer.music.play()
            lista_musicas.select_clear(0, tk.END)
            lista_musicas.selection_set(indice_anterior)
            lista_musicas.activate(indice_anterior)

# Cria uma lista de reprodução (Listbox) para mostrar as músicas.
lista_musicas = Listbox(janela, selectmode=tk.SINGLE, width=50)
lista_musicas.pack(padx=10, pady=10, side=tk.LEFT)

# Adiciona uma barra de rolagem à lista de reprodução.
scrollbar = Scrollbar(janela)
scrollbar.pack(fill=tk.Y, side=tk.RIGHT)
lista_musicas.config(yscrollcommand=scrollbar.set)
scrollbar.config(command=lista_musicas.yview)

# Cria botões estilizados para carregar, tocar, parar, avançar e voltar músicas.
botao_carregar = ttk.Button(janela, text="Carregar Música", command=carregar_musica)
botao_tocar = ttk.Button(janela, text="Tocar Música", command=tocar_musica)
botao_parar = ttk.Button(janela, text="Parar Música", command=parar_musica)
```

```
# Erro: Botões sem comandos
# Comentário: Os botões 'botao_avancar' e 'botao_voltar' foram removidos, mas suas referências continuam sendo usadas.

botao_avancar = ttk.Button(janela, text="Avançar")  # Erro: Falta de comando
botao_voltar = ttk.Button(janela, text="Voltar")  # Erro: Falta de comando

# Estiliza os botões.
estilo = ttk.Style()
estilo.configure("TButton", foreground="blue", background="white", font=("Helvetica", 12))
estilo.map("TButton",
        foreground=[("pressed", "red"), ("active", "blue")],
        background=[("pressed", "!disabled", "black"), ("active", "white")]
        )

# Inicia o loop principal do Pygame.
janela.mainloop()
```

3. DESENVOLVIMENTO

3.1. OBSERVAÇÃO 1: FALTA DE COMENTÁRIO E SEPARAÇÃO DE LINHAS

3.1.1. Correção 1: Falta de comentário e separação de linhas

Não é possível escrever uma linha de código com o nome do código junto a importação de sua biblioteca.

```
Python:import pygame
```

3.1.2. Correção 2: Falta de comentário

Caso deseje indicar o tipo de linguagem utilizada, deve-se comentar este em linha separada, podendo ser de duas maneiras possíveis.

```
# Python:
''' Python: '''
```

3.1.3. Opção de correção do erro: Separação de linhas

Após o devido comentário, o import pygame deve ser listado na linha logo após ou sobressaltando mais uma linha, criando um espaço entre a observação e o começo do código, estando o import pygame alinhado à esquerda do compilador.

```
# Linguagem Python:
import pygame
```

```
# Linguagem Python:

import pygame
```

3.2. OBSERVAÇÃO 2: INSTALAÇÃO E ATUALIZAÇÃO DA BIBLIOTECA PYGAME

3.2.1. Correção 1: Instalação da biblioteca pygame

Foi realizada tentativa de execução do código, com a biblioteca pygame.

```
import pygame
```

Mas como erro comum, este não foi executado, pois o módulo pygame não estava devidamente instalado ou podendo também como outra opção não estar disponível no ambiente Python em que se deseja executar o código. No caso

```
ModuleNotFoundError: No module named 'pygame'
```

```
ModuleNotFoundError: Nenhum módulo chamado 'pygame'
PS D:\Alunos\rb2023\TESTE DE SISTEMAS\TESTE_SISTEMAS>
```

Certifique-se de que esteja utilizando o ambiente Python correto, e que o pygame esteja devidamente instalado, utilizando o comando abaixo.

```
PS D:\Alunos\rb2023\TESTE DE SISTEMAS\TESTE_SISTEMAS> pip install pygame
```

Após a instalação no terminal, deverá ser verificado que ele foi instalado corretamente, conforme segue.

```
PS D:\Alunos\rb2023\TESTE DE SISTEMAS\TESTE_SISTEMAS> pip install pygame
Installing collected packages: pygame
Successfully installed pygame-2.5.2
```

3.2.2. Correção 2: Atualização da biblioteca pygame

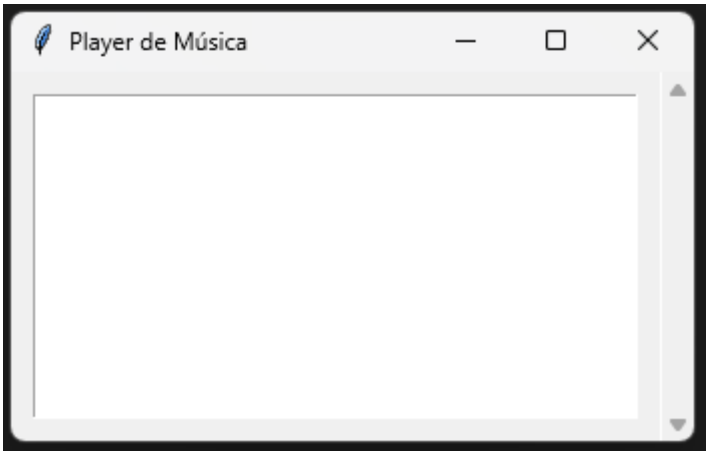
Também foi realizada a devida atualização, sendo esta sugerida para se evitar erros, conforme Após a instalação no terminal, deverá ser verificado que ele foi instalado corretamente, conforme segue.

```
PS D:\Alunos\rb2023\TESTE DE SISTEMAS\TESTE_SISTEMAS> python.exe -m pip install --upgrade pip
```

Sendo realizada de forma correta a atualização no terminal, deverá ser verificada a seguinte mensagem.

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pip in
```

3.3. OBSERVAÇÃO 3: FALTA DE COMANDOS E BOTÕES DE EXECUÇÃO



3.3.1. Correção 1: Falta de comandos de execução

```
# Cria botões estilizados para carregar, tocar, parar, avançar e voltar músicas.
botao_carregar = ttk.Button(janela, text="Carregar Música", command=carregar_musica)
botao_tocar = ttk.Button(janela, text="Tocar Música", command=tocar_musica)
botao_parar = ttk.Button(janela, text="Parar Música", command=parar_musica)

# Erro: Botões sem comandos
# Comentário: Os botões 'botao_avancar' e 'botao_voltar' foram removidos, mas suas referências continuam sendo usadas.

botao_avancar = ttk.Button(janela, text="Avançar")  # Erro: Falta de comando
botao_voltar = ttk.Button(janela, text="Voltar")  # Erro: Falta de comando
```

Foi verificado nas linhas de código acima, erros de comandos nos botões, estando estes sem comandos dos botões 'botao_avancar' e 'botao_voltar'. Sendo necessário a inserção de comandos corretos conforme abaixo, ficando no 'botao_avancar' o comando 'proxima_musica' e no 'botao_voltar' o comando 'musica_anterior'.

```
# Comandos dos botões para realizar o devido funcionamento do player
botao_carregar = ttk.Button(janela, text="Carregar Música", command=carregar_musica)
botao_tocar = ttk.Button(janela, text="Tocar Música", command=tocar_musica)
botao_parar = ttk.Button(janela, text="Parar Música", command=parar_musica)
```

```
botao_avancar = ttk.Button(janela, text="Avançar", command=proxima_musica)
botao_voltar = ttk.Button(janela, text="Voltar", command=musica_anterior)
```

3.3.2. Correção 2: Falta de botões de execução

```
# Não possuía as linhas de criação dos respectivos botões
```

Conforme verificado no código inicial, não existiam as linhas de comando para a devida criação dos botões de execução do player. Sendo necessária a criação de todos os botões para carregar, tocar, parar, avançar e voltar.

```
# Criando os botões de comando
botao_carregar.pack(padx=10, pady=10)
botao_tocar.pack(padx=10, pady=10)
botao_parar.pack(padx=10, pady=10)
botao_avancar.pack(padx=10, pady=10)
botao_voltar.pack(padx=10, pady=10)
```

Observação:

Como exemplo, na linha botao_carregar.pack(padx=10, pady=10), padx e pady são opções que controlam o preenchimento horizontal (esquerda e direita) e vertical (acima e abaixo) em pixels ao redor do botão quando ele é colocado na janela.

padx: Esta opção define o espaço de preenchimento horizontal (esquerda e direita) ao redor do widget, no caso, do botão botao_carregar. Com padx=10, haverá um espaço de 10 pixels à esquerda e à direita do botão, criando um espaço vazio ao seu redor.

pady: Esta opção define o espaço de preenchimento vertical (acima e abaixo) ao redor do widget. Com pady=10, haverá um espaço de 10 pixels acima e abaixo do botão.

3.4. OBSERVAÇÃO 4: POSSÍVEIS IMPLEMENTAÇÕES DE MELHORIAS

Foi constatado que o aplicativo oferece apenas a capacidade de carregar músicas individualmente, não permitindo o carregamento de todas de uma só vez.

3.4.1. Correção 1: Falta de comandos de execução

3.4.1.1. Código original

Conforme delimitado, nas linhas do código original, este somente executava o carregamento de uma música de cada vez.

```
# Função para carregar músicas.
def carregar_musicas():
        caminhos_do_arquivo = filedialog.askopenfilenames(title="Selecione músicas", filetypes=[("Arquivos de áudio", "*.mp3")])
    for caminho_do_arquivo in caminhos_do_arquivo:
```

```
# Comandos dos botões
botao_carregar = ttk.Button(janela, text="Carregar Música", command=carregar_musica) # Comando carregar música
```

```
def carregar_musica():
        caminho_do_arquivo = filedialog.askopenfilename(title="Selecione uma música", filetypes=[("Arquivos de áudio", "*.mp3")])
    if caminho_do_arquivo:
```

3.4.1.2. Código alterado

Após a implementação de melhorias, o código agora passou a ser capaz de carregar músicas de forma individual ou em lote, proporcionando ao usuário maior flexibilidade.

```
# Função para carregar todas as músicas de uma só vez.
def carregar_todas_musicas():
        caminhos_dos_arquivos = filedialog.askopenfilenames(title="Selecione músicas", filetypes=[("Arquivos de áudio", "*.mp3")])
    for caminho_do_arquivo in caminhos_dos_arquivos:
        lista_de_reproducao.append(caminho_do_arquivo)
        lista_musicas.insert(tk.END, caminho_do_arquivo)
```

```
# Comandos dos botões
botao_carregar = ttk.Button(janela, text="Carregar Música", command=carregar_musica) # Comando carregar uma música
botao_carregar_todas = ttk.Button(janela, text="Carregar Todas as Músicas", command=carregar_todas_musicas) # Carregar todas as músicas
```

```
botao_carregar_todas.pack(padx=10, pady=10)
```

4. APLICATIVO COM ALTERAÇÕES (sem BUGS e com melhorias)

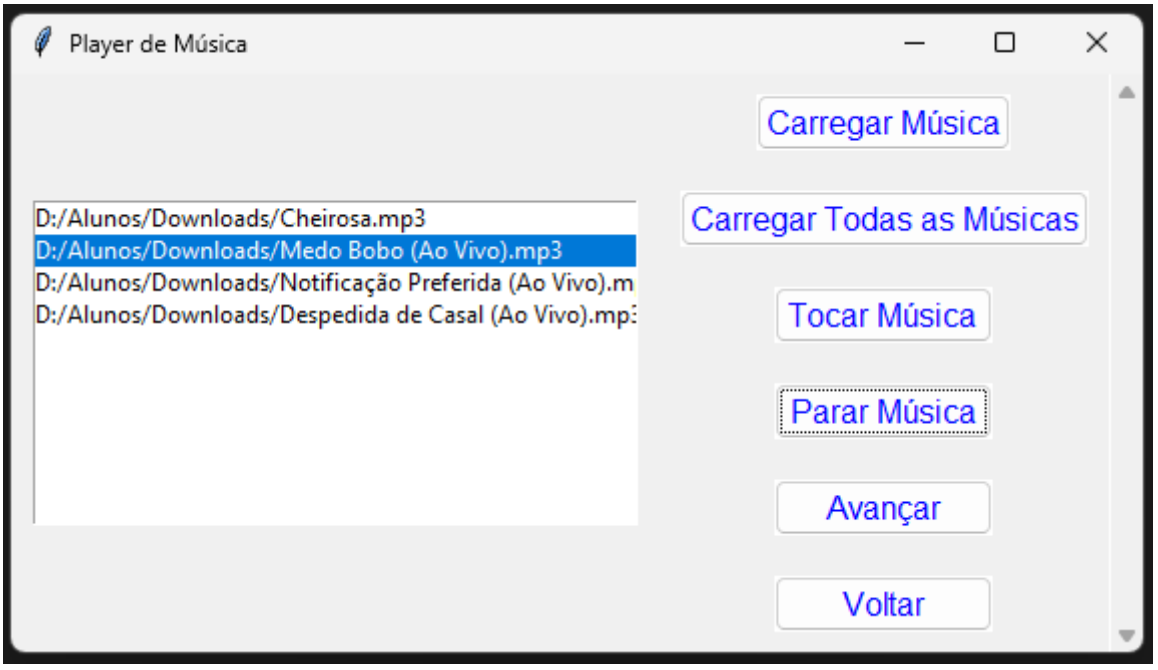
```
import pygame
```

```
import tkinter as tk
from tkinter import filedialog
from tkinter import Listbox, Scrollbar, ttk

# Inicializa o Pygame.
pygame.init()

# Cria uma janela.
janela = tk.Tk()
janela.title("Player de Música")

# Cria uma lista para armazenar as músicas.
lista_de_reproducao = []

# Função para carregar uma única música.
def carregar_musica():
        caminho_do_arquivo = filedialog.askopenfilename(title="Selecione uma música",
filetypes=[("Arquivos de áudio", "*.mp3")])
    if caminho_do_arquivo:
        lista_de_reproducao.append(caminho_do_arquivo)
        lista_musicas.insert(tk.END, caminho_do_arquivo)

# Função para carregar todas as músicas de uma só vez.
def carregar_todas_musicas():
        caminhos_dos_arquivos = filedialog.askopenfilenames(title="Selecione músicas",
filetypes=[("Arquivos de áudio", "*.mp3")])
    for caminho_do_arquivo in caminhos_dos_arquivos:
        lista_de_reproducao.append(caminho_do_arquivo)
        lista_musicas.insert(tk.END, caminho_do_arquivo)

# Função para tocar a música selecionada na lista.
def tocar_musica():
```

```
    indice_selecionado = lista_musicas.curselection()
    if indice_selecionado:
        musica_selecionada = lista_de_reproducao[indice_selecionado[0]]
        pygame.mixer.music.load(musica_selecionada)
        pygame.mixer.music.play()

# Função para interromper a música atual.
def parar_musica():
    pygame.mixer.music.stop()

# Função para avançar para a próxima música.
def proxima_musica():
    indice_atual = lista_musicas.curselection()
    if indice_atual:
        indice_proxima = indice_atual[0] + 1
        if indice_proxima < len(lista_de_reproducao):
            musica_selecionada = lista_de_reproducao[indice_proxima]
            pygame.mixer.music.load(musica_selecionada)
            pygame.mixer.music.play()
            lista_musicas.select_clear(0, tk.END)
            lista_musicas.selection_set(indice_proxima)
            lista_musicas.activate(indice_proxima)

# Função para voltar para a música anterior.
def musica_anterior():
    indice_atual = lista_musicas.curselection()
    if indice_atual:
        indice_anterior = indice_atual[0] - 1
        if indice_anterior >= 0:
            musica_selecionada = lista_de_reproducao[indice_anterior]
            pygame.mixer.music.load(musica_selecionada)
            pygame.mixer.music.play()
```

```
            lista_musicas.select_clear(0, tk.END)
            lista_musicas.selection_set(indice_anterior)
            lista_musicas.activate(indice_anterior)

# Cria uma lista de reprodução (Listbox) para mostrar as músicas.
lista_musicas = Listbox(janela, selectmode=tk.SINGLE, width=50)
lista_musicas.pack(padx=10, pady=10, side=tk.LEFT)

# Adiciona uma barra de rolagem à lista de reprodução.
scrollbar = Scrollbar(janela)
scrollbar.pack(fill=tk.Y, side=tk.RIGHT)
lista_musicas.config(yscrollcommand=scrollbar.set)
scrollbar.config(command=lista_musicas.yview)

# Cria botões/comandos estilizados para carregar, tocar, parar, avançar e voltar músicas.

# Comandos dos botões
botao_carregar = ttk.Button(janela, text="Carregar Música", command=carregar_musica) # Comando carregar uma música
botao_carregar_todas = ttk.Button(janela, text="Carregar Todas as Músicas", command=carregar_todas_musicas) # Carregar todas as músicas
botao_tocar = ttk.Button(janela, text="Tocar Música", command=tocar_musica)
botao_parar = ttk.Button(janela, text="Parar Música", command=parar_musica)
botao_avancar = ttk.Button(janela, text="Avançar", command=proxima_musica)
botao_voltar = ttk.Button(janela, text="Voltar", command=musica_anterior)

# Criando os botões de comando
botao_carregar.pack(padx=10, pady=10)
botao_carregar_todas.pack(padx=10, pady=10)
botao_tocar.pack(padx=10, pady=10)
botao_parar.pack(padx=10, pady=10)
botao_avancar.pack(padx=10, pady=10)
```

```
botao_voltar.pack(padx=10, pady=10)

# Estiliza os botões.
estilo = ttk.Style()
estilo.configure("TButton", foreground="blue", background="white", font=("Helvetica", 12))
estilo.map("TButton",
        foreground=[("pressed", "red"), ("active", "blue")],
        background=[("pressed", "!disabled", "black"), ("active", "white")]
        )

# Inicia o loop principal do Pygame.
janela.mainloop()
```



5. CONCLUSÃO

Neste processo de análise e correção do código do aplicativo de música em Python, várias questões foram abordadas para tornar o código mais claro, funcional e livre de erros. Foi destacado que a organização e a documentação são partes essenciais do desenvolvimento de qualquer programa.

A introdução destacou a importância dos aplicativos de música na vida moderna, destacando como eles se tornaram parte integrante de nossas vidas. O desenvolvimento da correção incluiu a resolução de erros e a melhoria da legibilidade do código.

Conclui-se assim que, as correções e melhorias são cruciais para garantir que o aplicativo de música funcione corretamente e proporcione uma experiência positiva ao usuário, pois no desenvolvimento de aplicativos, a atenção aos detalhes e a manutenção constante são fundamentais para garantir que o software atenda às necessidades e expectativas dos usuários. Portanto, ao criar ou corrigir aplicativos, é importante realizar testes extensivos e garantir que todas as funcionalidades estejam operando conforme o esperado.