Atividade Prática: Implementar Módulo de Cadastro de Membros

Objetivo da Atividade

Desenvolver uma interface para cadastro e visualização de membros usando CustomTkinter, seguindo o mesmo padrão arquitetural do módulo de livros existente.

📋 TAREFA 1: Configuração Inicial e Importações

O que fazer:
Configurar o ambiente com todas as importações necessárias e estabelecer a conexão com o banco de dados.

Como fazer:

```
python
# IMPORTANTE: Estas importações já existem no código original
# Você apenas precisa garantir que estão disponíveis

from conexaoDB import ConexaoDB
import dotenv
import os
import customtkinter as ctk
from tkinter import ttk, messagebox

# Carregar variáveis de ambiente
dotenv.load_dotenv(dotenv.find_dotenv())

# Configurar parâmetros do banco
DB_NAME = os.getenv("DB_NAME")
DB_USER = os.getenv("DB_USER")
DB_PASSWORD = os.getenv("DB_PASSWORD")
DB_HOST = os.getenv("DB_HOST")
DB_PORT = os.getenv("DB_PORT")
```

```
# Criar instância do banco (JÁ EXISTE NO CÓDIGO ORIGINAL)
meuBanco = ConexaoDB(DB_NAME, DB_HOST, DB_PORT, DB_USER, DB_PASSWORD)
```

Dica prática:

- Estas configurações já estão presentes no código do professor
- O objeto `meuBanco` já está criado e funcionando
- Você só precisa usar `meuBanco.manipular()` e `meuBanco.consultar()` em suas funções

📋 TAREFA 2: Analisar a Estrutura Existente e Planejar

O que fazer:
Estude atentamente o código do cadastro de livros para entender:

- Como os frames são organizados
- Como as funções são estruturadas
- Como a conexão com banco funciona
- Como a tabela é atualizada

Dica prática:

- Observe que há dois componentes principais: formulário (Frame superior) e tabela (Frame inferior)
- Note que cada campo tem seu label e input organizados em grid
- Perceba que as funções seguem um padrão: validação → inserção → feedback → limpeza → atualização

Resultado esperado: Compreensão clara de como replicar a estrutura para membros.

📋 TAREFA 3: Criar a Função de Cadastro de Membros

O que fazer:
Implementar a função `cadastrar_membro()` que será chamada pelo botão de cadastro.

Como fazer:
Siga este esqueleto baseado no padrão existente:

```
python
def cadastrar_membro():
    # 1. Obter valores dos campos (similar a campos de livro)
    nome = campo_nome_membro.get().strip()
    email = campo_email_membro.get().strip()

    # 2. Validar se nome não está vazio (como no título do livro)
    if not nome:
        messagebox.showerror("ERRO DE VALIDAÇÃO", "Nome é obrigatório!")
        campo_nome_membro.focus_set()
        return

    # 3. Validar se email não está vazio (como no ano do livro)
    if not email:
        messagebox.showerror("ERRO DE VALIDAÇÃO", "Email é obrigatório!")
        campo_email_membro.focus_set()
        return

    # 4. Se tudo válido, inserir no banco (usando meuBanco.manipular)
    meuBanco.manipular('''
        INSERT INTO membros
        VALUES (default, %s, %s);
    ''', [nome, email])

    # 5. Limpar campos (como em cadastrar_livro)
    campo_nome_membro.delete(0, 'end')
    campo_email_membro.delete(0, 'end')

    # 6. Mostrar mensagem de sucesso
    messagebox.showinfo("CADASTRADO COM SUCESSO", "MEMBRO CADASTRADO COM SUCESSO")

    # 7. Atualizar a tabela (DESAFIO - similar ao que precisa ser feito em livros)
    # Implementaremos isso na Tarefa 7
```

Dica prática:

- Use `meuBanco.manipular()` para INSERT, UPDATE, DELETE
- Use `meuBanco.consultar()` para SELECT (veremos na próxima tarefa)

- As validações devem seguir o mesmo padrão visual (messagebox e focus_set)

📋 TAREFA 4: Implementar a Interface do Formulário

O que fazer:
Criar o frame do formulário com campos para nome e email, seguindo o mesmo layout do formulário de livros.

Como fazer:

```
python
# Container principal do formulário
formulario_membro = ctk.CTkFrame(janela)
formulario_membro.pack(fill="x", padx=10, pady=10)

# Configure grid (2 colunas como no original)
formulario_membro.columnconfigure(0, weight=0, minsize=120)
formulario_membro.columnconfigure(1, weight=1)

# Título do formulário
titulo_membro = ctk.CTkLabel(
    formulario_membro,
    text="CADASTRAR MEMBRO",
    font=ctk.CTkFont(size=24, weight="bold")
)
titulo_membro.grid(row=0, column=0, columnspan=2, pady=(0, 10))

# Campo Nome
label_nome = ctk.CTkLabel(formulario_membro, text="NOME:")
label_nome.grid(row=1, column=0, sticky="w", padx=5, pady=5)

campo_nome_membro = ctk.CTkEntry(
    formulario_membro,
    width=300,
    placeholder_text="Digite o nome do membro:"
)
campo_nome_membro.grid(row=1, column=1, sticky="ew", padx=5, pady=5)

# Campo Email
label_email = ctk.CTkLabel(formulario_membro, text="EMAIL:")
label_email.grid(row=2, column=0, sticky="w", padx=5, pady=5)
```

```
campo_email_membro = ctk.CTkEntry(
    formulario_membro,
    width=300,
    placeholder_text="Digite o email do membro:"
)
campo_email_membro.grid(row=2, column=1, sticky="ew", padx=5, pady=5)

# Botão Cadastrar
botao_cadastrar_membro = ctk.CTkButton(
    formulario_membro,
    text="Cadastrar Membro",
    command=cadastrar_membro  # Conecta com a função da Tarefa 3
)
botao_cadastrar_membro.grid(row=3, column=1, sticky="e", padx=5, pady=5)
```

Dica prática:

- Use o mesmo sistema de grid do formulário de livros
- Mantenha consistência nos paddings e margens
- Conecte o botão à função `cadastrar_membro` usando `command=`

📋 TAREFA 5: Criar a Tabela de Visualização

O que fazer:
Implementar uma tabela (Treeview) para exibir os membros cadastrados.

Como fazer:

```
python
# Container da tabela
container_tabela_membros = ctk.CTkFrame(janela)
container_tabela_membros.pack(fill="both", expand=True, padx=10, pady=(0, 10))

# Definir colunas da tabela
colunas_membros = ["ID", "Nome", "Email"]

# Criar a tabela
tabela_membros = ttk.Treeview(
    container_tabela_membros,
    columns=colunas_membros,
```

```
    show="headings",
    height=10
)

# Configurar cabeçalhos
tabela_membros.heading("ID", text="ID Membro")
tabela_membros.heading("Nome", text="Nome")
tabela_membros.heading("Email", text="Email")

# Configurar larguras das colunas
tabela_membros.column("ID", width=50)
tabela_membros.column("Nome", width=200)
tabela_membros.column("Email", width=250)

# Empacotar a tabela
tabela_membros.pack(fill="both", expand=True)
```

Dica prática:

- Siga exatamente o padrão da tabela de livros
- A altura (height) pode ser ajustada conforme necessidade
- As larguras das colunas devem ser proporcionais ao conteúdo esperado

---

📋 TAREFA 6: Implementar Carregamento de Dados na Tabela

O que fazer:
Criar a função `carregar_membros()` que busca dados do banco e preenche a tabela.

Como fazer:

```
python
def carregar_membros():
    # Limpar tabela atual (importante para evitar duplicatas)
    for item in tabela_membros.get_children():
        tabela_membros.delete(item)

    # Consultar membros no banco
    membros = meuBanco.consultar('''
```

```
        SELECT id_membro, nome_membro, email_membro
        FROM membros
        ORDER BY id_membro ASC;
    ''', [])

    # Inserir cada membro na tabela
    for membro in membros:
        tabela_membros.insert("", "end", values=membro)
```

Dica prática:

- Sempre limpe a tabela antes de recarregar para evitar dados duplicados
- A ordem das colunas no SELECT deve bater com a ordem na tabela
- Use `ORDER BY` para manter uma ordenação consistente

📋 TAREFA 7: Conectar o Cadastro com a Atualização da Tabela

O que fazer:
Fazer com que a tabela seja atualizada automaticamente após cadastrar um novo membro.

Como fazer:
Modifique a função `cadastrar_membro()` da Tarefa 3, adicionando no final:

```
python
# No final da função cadastrar_membro(), após mostrar mensagem de sucesso:
carregar_membros()  # Atualiza a tabela com o novo membro
```

Dica prática:

- Esta chamada deve ser a última coisa na função, após a inserção bem-sucedida
- Isso garante que a tabela sempre mostre os dados mais recentes

---

📋 TAREFA 8: Integrar e Testar o Sistema Completo

O que fazer:
Garantir que todos os componentes funcionem juntos corretamente.

Checklist de teste:

- Formulário aparece na interface
- Campos de nome e email são editáveis
- Botão cadastrar está visível e clicável
- Tabela de membros aparece abaixo do formulário
- Ao cadastrar membro: dados salvos no banco
- Ao cadastrar membro: campos são limpos
- Ao cadastrar membro: mensagem de sucesso aparece
- Ao cadastrar membro: novo membro aparece na tabela
- Validação funciona (campos vazios mostram erro)

Como testar:

1. Execute a aplicação
2. Preencha o formulário com nome e email
3. Clique em "Cadastrar Membro"
4. Verifique se o membro aparece na tabela
5. Teste as validações tentando cadastrar sem nome ou sem email

Dica prática:

- Teste com dados reais
- Verifique se há erros no console
- Confirme no banco se os dados estão sendo salvos

🎯 ENTREGÁVEL FINAL

Ao concluir todas as tarefas, você deve ter:

✅ Uma interface com formulário de cadastro de membros
✅ Uma tabela que exibe membros cadastrados

✅ Funcionalidade completa de cadastro e visualização
✅ Validações de campos obrigatórios
✅ Atualização automática da tabela