Gerenciador de Clientes - Documentação

Bem-vindo ao **Gerenciador de Clientes**, uma aplicação simples em Python que ajuda a gerenciar informações de clientes (como nome, sobrenome, email e CPF) usando um banco de dados SQLite. Este projeto foi usando **Python**, **SQL** e **SQLite**, com uma interface gráfica feita com **Tkinter**. Ele é perfeito para entender como criar uma aplicação com interface, conectar com um banco de dados e realizar operações como adicionar, visualizar, buscar, atualizar e deletar registros.

O que faz esta aplicação?

A aplicação permite:

- **Adicionar** novos clientes ao banco de dados.
- **Visualizar** todos os clientes em uma tabela.
- **Buscar** clientes por nome, sobrenome, email ou CPF.
- **Atualizar** os dados de um cliente existente.
- **Deletar** clientes do banco de dados.

A interface gráfica é fácil de usar, com campos para inserir dados e botões para executar as ações. O banco de dados SQLite armazena os dados em um arquivo chamado clientes.db.

Estrutura do Projeto

O projeto é dividido em três arquivos Python para manter o código organizado e fácil de entender:

1. **Gui.py**:
    - Contém a classe Gui, que cria a interface gráfica usando Tkinter.
    - Inclui campos de texto para inserir dados do cliente, botões para ações (Adicionar, Atualizar, Deletar, Buscar, Limpar) e uma tabela para mostrar os clientes.
    - Conecta-se à classe Backend para realizar operações no banco de dados.
2. **Backend.py**:
    - Contém a classe Backend, que gerencia todas as operações do banco de dados SQLite.
    - Inclui métodos para conectar ao banco, criar a tabela, inserir, visualizar, buscar, atualizar e deletar clientes.
    - Usa consultas SQL seguras para evitar problemas de segurança.
3. **application.py**:
    - O arquivo principal que inicia a aplicação.
    - Inicializa o banco de dados e abre a janela gráfica.

Pré-requisitos

Para executar este projeto, você precisa de:

- **Python 3.x** instalado (recomendado: Python 3.8 ou superior). Você pode baixar em python.org.
- **Tkinter**: Já vem incluído com o Python, então você não precisa instalar nada extra.
- **SQLite**: Também incluído no Python (módulo sqlite3), sem necessidade de instalação adicional.
- **PyInstaller** (opcional): Necessário apenas se você quiser criar um arquivo executável. Veja a seção "Criando um Executável" abaixo.

Como Configurar e Executar

Siga estas etapas para rodar a aplicação:

1. **Baixe ou crie os arquivos**:
    - Crie uma pasta para o projeto (exemplo: gerenciador_clientes).
    - Salve os três arquivos (Gui.py, Backend.py, application.py) nessa pasta. Você pode copiar os códigos fornecidos pelo instrutor.
2. **Verifique se o Python está instalado**:
    - Abra um terminal (Prompt de Comando no Windows, Terminal no Linux/Mac) e digite:
    - python --version
    - Se o Python estiver instalado, você verá a versão (exemplo: Python 3.9.2). Caso contrário, baixe e instale o Python.
3. **Execute a aplicação**:
    - No terminal, navegue até a pasta do projeto:
    - cd caminho/para/gerenciador_clientes
    - Execute o arquivo principal:
    - python application.py
    - Uma janela gráfica será aberta, mostrando a interface do Gerenciador de Clientes.

Como Usar a Aplicação

Quando você executa application.py, uma janela aparece com:

- **Campos de entrada**: Para inserir Nome, Sobrenome, Email e CPF.
- **Botões**:
    - **Adicionar**: Insere um novo cliente no banco de dados.

- **Atualizar**: Atualiza os dados do cliente selecionado na tabela.
- **Deletar**: Remove o cliente selecionado.
- **Buscar**: Filtra a tabela com base nos dados inseridos nos campos.
- **Limpar**: Limpa os campos de entrada.

- **Tabela (Treeview)**: Mostra todos os clientes do banco de dados.

Passo a passo:

1. **Adicionar um cliente**:
    - Preencha os campos Nome, Sobrenome, Email e CPF.
    - Clique em "Adicionar".
    - Uma mensagem de sucesso aparecerá, e a tabela será atualizada.
2. **Visualizar clientes**:
    - A tabela mostra automaticamente todos os clientes ao abrir a aplicação.
3. **Buscar clientes**:
    - Insira um valor em qualquer campo (exemplo: Nome ou CPF) e clique em "Buscar".
    - A tabela mostrará apenas os clientes que correspondem aos critérios.
4. **Atualizar um cliente**:
    - Clique em um cliente na tabela (os campos serão preenchidos automaticamente).
    - Edite os campos desejados e clique em "Atualizar".
    - A tabela será atualizada com as novas informações.
5. **Deletar um cliente**:
    - Clique em um cliente na tabela e clique em "Deletar".
    - O cliente será removido, e a tabela será atualizada.

Criando um Executável com PyInstaller

Para facilitar o uso da aplicação sem precisar executar o Python no terminal, você pode criar um arquivo executável (.exe no Windows, ou equivalente em outros sistemas) usando o **PyInstaller**.

Passos para criar o executável:

1. **Instale o PyInstaller**:
    - No terminal, execute:
    - pip install pyinstaller

2. **Crie o executável**:
    - Navegue até a pasta do projeto no terminal:
    - cd caminho/para/gerenciador_clientes
    - Execute o comando abaixo para criar um executável a partir de application.py:
    - pyinstaller --onefile application.py
    - Explicação:
        - --onefile: Gera um único arquivo executável (mais fácil de compartilhar).
        - O comando cria uma pasta dist na sua pasta do projeto, contendo o arquivo executável (application.exe no Windows).
3. **Execute o executável**:
    - Vá para a pasta dist (exemplo: gerenciador_clientes/dist).
    - Clique duas vezes no arquivo application.exe (ou execute pelo terminal).
    - A aplicação abrirá como antes, mas sem precisar do Python instalado no computador.
4. **Dica**:
    - O executável pode ser grande (cerca de 50-100 MB) porque inclui o Python e todas as dependências.
    - Para compartilhar, copie o arquivo .exe (e o arquivo clientes.db, se já tiver dados) para outros computadores.

Estrutura do Banco de Dados

O banco de dados SQLite é salvo em um arquivo chamado clientes.db na mesma pasta do projeto. A tabela clientes tem a seguinte estrutura:

- **id**: Um número único para cada cliente (gerado automaticamente).
- **nome**: O nome do cliente (texto).
- **sobrenome**: O sobrenome do cliente (texto).
- **email**: O email do cliente (texto).
- **cpf**: O CPF do cliente (texto).

O método Backend.initDB() cria essa tabela automaticamente na primeira execução.

Dicas para Solução de Problemas

- **Erro: "No module named tkinter"**:

    - Certifique-se de que o Python está instalado corretamente. Tkinter vem com o Python, mas pode estar faltando em algumas instalações. Reinstale o Python ou instale Tkinter:
    - pip install tk
- **Erro ao executar o programa**:
    - Verifique se todos os arquivos (Gui.py, Backend.py, application.py) estão na mesma pasta.
    - Confirme que você está executando python application.py na pasta correta.
- **O executável não abre**:
    - Certifique-se de que o PyInstaller foi executado com sucesso.
    - No Windows, tente executar o .exe pelo terminal para ver mensagens de erro:
    - ./dist/application.exe
- **A tabela não atualiza**:
    - Certifique-se de que o arquivo clientes.db está na mesma pasta do executável ou dos arquivos Python.

Para Alunos: O que você pode aprender com este projeto?

- **Python**: Como criar classes, métodos estáticos, e organizar código em módulos.
- **SQL e SQLite**: Como criar tabelas, inserir, consultar, atualizar e deletar dados usando consultas SQL seguras.
- **Tkinter**: Como criar uma interface gráfica com campos de texto, botões e tabelas.
- **Boas práticas**: Uso de consultas parametrizadas para segurança, separação de responsabilidades (GUI vs. Backend), e comentários claros.
- **PyInstaller**: Como transformar um programa Python em um executável para facilitar a distribuição.

Próximos Passos

Experimente adicionar novas funcionalidades, como:

- Validação de CPF para garantir que o formato está correto.
- Exportar a lista de clientes para um arquivo CSV.
- Adicionar um botão para recarregar todos os clientes após uma busca.
- Melhorar a interface com cores ou ícones.