Documentação do Código

Arquivo: main.py

Descrição Geral:

Este arquivo contém a implementação de uma aplicação gráfica desenvolvida em Python usando Tkinter para gerenciar usuários em um banco de dados. A aplicação permite adicionar, atualizar, excluir, e exibir informações de usuários.

Estrutura do Código:

1 - Importações Necessárias:

- tkinteretkinter.simpledialog : Fornece ferramentas para criar uma interface gráfica.
- tkinter.messagebox: Exibe mensagens de alerta ou confirmação.
- DatabaseManager: Classe do arquivo db_manager.py usado para interagir com o banco de dados.

2- ClasseApp:

- Implementa uma interface gráfica da aplicação e integração com o banco de dados.
- Métodos principais:
    - __init__: Configura a janela principal e inicializa o banco de dados.
    - setup_ui: Monta uma interface gráfica com campos de entrada, botões e uma lista de usuários.
    - add_user: Adiciona um novo usuário ao banco de dados.
    - update_user: Atualiza um campo específico de um usuário no banco.
    - delete_user: Remove um usuário com base no ID.
    - load_user_data: Carrega os dados do usuário selecionado na lista para os campos de entrada.
    - display_users: Exibe os usuários cadastrados no banco de dados na lista.
    - clear_entries: Limpa os campos de entrada.

3 - Execução Principal:

- A aplicação é iniciada com a criação de uma instância da classe App e a chamada do loop principal do Tkinter (root.mainloop()).

Funcionalidades Sucesso:

Interfáce Gráfica:

- Campos de Entrada:
  - Nome, E-mail, Idade, CPF, CEP e Gênero.
- Botões:
  - Adicionar Usuário: Inserir um novo usuário no banco.
  - Atualizar Usuário: Atualiza os dados de um usuário específico.
  - Deletar Usuário: Exclui um usuário pelo ID.
- Lista de usuários:
  - Mostra informações como RG, Nome, E-mail, Idade, CPF, CEP, e Gênero.
  - Permite selecionar um usuário para edição.

Arquivo: db_manager.py

Descrição Geral:

Este arquivo contém a classe DatabaseManager, que gerencia a conexão e as operações realizadas no banco de dados. Ele serve como intermediário entre o banco de dados e a aplicação gráfica.

Estrutura do Código:

1 - Importações Necessárias:

- mysql.connector: Biblioteca para conectar e executar comandos em um banco de dados MySQL.

2 - ClasseDatabaseManager:

- Atributos**:**
  - config: Dicionário que contém informações de configuração da conexão.
  - connection: Objeto que gerencia a conexão ativa com o banco.
- Métodos**:**
  - __init__: Inicializa a conexão com o banco e garante que existe uma tabela de usuários.
  - create_connection: Estabelece a conexão com o banco usando as configurações fornecidas.
  - create_table: Cria uma tabela users se ela não existir.
  - insert_user: Insere um novo registro na tabela users.
  - fetch_users: Recupera todos os registros da tabela users.

- update_user_field: Atualiza um campo específico de um registro na tabela users.
- delete_user: Remove um registro da tabela userspelo ID.
- __del__: Garante que a conexão com o banco esteja fechada ao destruir uma instância.

Funcionalidades Sucesso:

Tabela: Users

Campos:

- id(INT, Incremento Automático, Chave Primária)
- name(VARCHAR(255), Obrigatório)
- email(VARCHAR(255), Único, Obrigatório)
- age(INT, Obrigatório)
- cpf(VARCHAR(14), Único, Obrigatório)
- cep(VARCHAR(9), Obrigatório)
- gender(VARCHAR(20), Obrigatório)

Fluxo de Operações:

1 - Conexão com o Banco:

- A conexão é criada no início da execução e permanece ativa até a aplicação ser encerrada.

2 - Manipulação de Dados:

- Inserção (insert_user): Recebe os dados do usuário como parâmetros e insere na tabela.
- Consulta (fetch_users): Recupera todos os registros e retorna como uma lista de tuplas.
- Atualização (update_user_field): Atualiza um campo específico de um usuário com base no ID.
- Remoção (delete_user): Exclui o registro de um usuário com base no ID.

3 - Encerramento:

- A conexão com o banco de dados é encerrada automaticamente quando uma instância de dados `DatabaseManager`é destruída.

Exemplo de uso:

1. O usuário insere os dados na interface gráfica e clica em "Adicionar Usuário".
2. O método add_user na classe App valida os dados e chama insert_user no DatabaseManager.
3. O banco de dados armazena os dados na tabela users.
4. A lista na interface é atualizada para refletir o novo registro.