Gerenciador de Clientes – Documentação Técnica

O Gerenciador de Clientes é uma aplicação desktop desenvolvida em Python que realiza o gerenciamento de registros de clientes, incluindo atributos como nome, sobrenome, email e CPF, utilizando banco de dados SQLite para persistência. A interface gráfica é implementada com Tkinter, oferecendo funcionalidades para operações CRUD (Create, Read, Update, Delete).

- Funcionalidades da Aplicação:
- Inserção de novos registros de clientes no banco de dados.
- Visualização de todos os clientes através de uma tabela dinâmica.
- Busca de clientes por campos: nome, sobrenome, email ou CPF.
- Atualização dos dados existentes de um cliente selecionado.
- Exclusão de clientes do banco de dados.

Arquitetura do Projeto:

Gui.py: Define a classe Gui, responsável pela construção da interface gráfica com Tkinter. Inclui campos de entrada, botões para operações (Adicionar, Atualizar, Deletar, Buscar, Limpar) e componente de exibição (Treeview) para listagem dos clientes. Integração com o módulo Backend para execução das operações no banco de dados.

Backend.py: Contém a classe Backend com métodos para conexão com banco SQLite, criação da tabela clientes, e execução de operações SQL para inserção, consulta, atualização e exclusão. As consultas utilizam parâmetros para prevenção de injeção SQL.

application.py: Arquivo principal responsável por iniciar a aplicação, realizando a inicialização do banco de dados via Backend.initDB() e instanciando a interface gráfica, além de executar o loop principal do Tkinter.

Requisitos para Execução:

Python 3.x (versão 3.8 ou superior recomendada).

Biblioteca Tkinter (inclusa na distribuição padrão do Python).

Módulo sqlite3 (incluso no Python).

Opcional: PyInstaller para geração de executável standalone.

Procedimento para Execução:

Colocar os arquivos Gui.py, Backend.py e application.py no mesmo diretório.

Executar via terminal o comando: python application.py.

A interface gráfica será exibida permitindo interação do usuário com as funcionalidades descritas.

Estrutura da Interface Gráfica:

Campos de entrada: nome, sobrenome, email, cpf.

Botões operacionais: Adicionar, Atualizar, Deletar, Buscar, Limpar.

Tabela de visualização dos registros, atualizada dinamicamente conforme operações realizadas.

Banco de Dados SQLite:

Arquivo: clientes.db localizado na raiz do projeto.

Tabela clientes com os seguintes campos:

id: INTEGER PRIMARY KEY AUTOINCREMENT

nome: TEXT NOT NULL

sobrenome: TEXT NOT NULL

email: TEXT NOT NULL

cpf: TEXT NOT NULL

Gerenciamento do Banco de Dados:

Método Backend.initDB() garante a criação da tabela na primeira execução.

Operações de inserção, atualização, exclusão e consulta realizadas via métodos específicos da classe Backend, utilizando comandos SQL parametrizados.

Distribuição e Criação de Executável:

Uso do PyInstaller para empacotamento da aplicação em arquivo executável único.

Comando para geração do executável:

pyinstaller --onefile application.py

Executável gerado na pasta dist/, podendo ser distribuído sem necessidade de instalação prévia do Python.

Considerações Adicionais:

Tratamento de erros básicos, como verificação de preenchimento dos campos antes da execução de operações.

Mensagens de feedback ao usuário após cada operação (sucesso ou falha).

Sugestões para aprimoramento incluem validação formal do CPF, exportação de dados e melhorias na interface.

Gerenciador de Clientes – Documentação Técnica

O Gerenciador de Clientes é uma aplicação desktop desenvolvida em Python que realiza o gerenciamento de registros de clientes, incluindo atributos como nome, sobrenome, email e CPF, utilizando banco de dados SQLite para persistência. A interface gráfica é implementada com Tkinter, oferecendo funcionalidades para operações CRUD (Create, Read, Update, Delete).

- Funcionalidades da Aplicação:
- Inserção de novos registros de clientes no banco de dados.
- Visualização de todos os clientes através de uma tabela dinâmica.
- Busca de clientes por campos: nome, sobrenome, email ou CPF.
- Atualização dos dados existentes de um cliente selecionado.
- Exclusão de clientes do banco de dados.

Arquitetura do Projeto:

Gui.py: Define a classe Gui, responsável pela construção da interface gráfica com Tkinter. Inclui campos de entrada, botões para operações (Adicionar, Atualizar, Deletar, Buscar, Limpar) e componente de exibição (Treeview) para listagem dos clientes. Integração com o módulo Backend para execução das operações no banco de dados.

Backend.py: Contém a classe Backend com métodos para conexão com banco SQLite, criação da tabela clientes, e execução de operações SQL para inserção, consulta, atualização e exclusão. As consultas utilizam parâmetros para prevenção de injeção SQL.

application.py: Arquivo principal responsável por iniciar a aplicação, realizando a inicialização do banco de dados via Backend.initDB() e instanciando a interface gráfica, além de executar o loop principal do Tkinter.

Requisitos para Execução:

Python 3.x (versão 3.8 ou superior recomendada).

Biblioteca Tkinter (inclusa na distribuição padrão do Python).

Módulo sqlite3 (incluso no Python).

Opcional: PyInstaller para geração de executável standalone.

Procedimento para Execução:

Colocar os arquivos Gui.py, Backend.py e application.py no mesmo diretório.

Executar via terminal o comando: python application.py.

A interface gráfica será exibida permitindo interação do usuário com as funcionalidades descritas.

Estrutura da Interface Gráfica:

Campos de entrada: nome, sobrenome, email, cpf.

Botões operacionais: Adicionar, Atualizar, Deletar, Buscar, Limpar.

Tabela de visualização dos registros, atualizada dinamicamente conforme operações realizadas.

Banco de Dados SQLite:

Arquivo: clientes.db localizado na raiz do projeto.

Tabela clientes com os seguintes campos:

id: INTEGER PRIMARY KEY AUTOINCREMENT

nome: TEXT NOT NULL

sobrenome: TEXT NOT NULL

email: TEXT NOT NULL

cpf: TEXT NOT NULL

Gerenciamento do Banco de Dados:

Método Backend.initDB() garante a criação da tabela na primeira execução.

Operações de inserção, atualização, exclusão e consulta realizadas via métodos específicos da classe Backend, utilizando comandos SQL parametrizados.

Distribuição e Criação de Executável:

Uso do PyInstaller para empacotamento da aplicação em arquivo executável único.

Comando para geração do executável:

pyinstaller --onefile application.py

Executável gerado na pasta dist/, podendo ser distribuído sem necessidade de instalação prévia do Python.

Considerações Adicionais:

Tratamento de erros básicos, como verificação de preenchimento dos campos antes da execução de operações.

Mensagens de feedback ao usuário após cada operação (sucesso ou falha).

Sugestões para aprimoramento incluem validação formal do CPF, exportação de dados e melhorias na interface.