


Exercício: Cadastro de Clientes – Etapas Separadas


Etapa 1 – Criar a Tabela de Clientes

 Arquivo: `estrutura.sql`


- Adicione um comando que crie uma nova tabela chamada `clientes`.
 - Essa tabela deve conter os seguintes campos:
 - Um campo `id` que será chave primária e auto incremento.
 - Um campo `nome` com limite de 50 caracteres.
 - Um campo `cpf` com exatamente 14 caracteres.
 - Execute o arquivo `.sql` no seu banco de dados para garantir que a tabela foi criada corretamente.
-

Etapa 2 – Cadastrar Cliente (INSERT)

 Arquivo: `cliente_repositorio.py`

 Local: `src/repositorios/`


1. Crie a função `cadastrar(nome_cliente: str, cpf_cliente: str)`.
 - Essa função deve abrir uma conexão com o banco de dados.
 - Criar um cursor.
 - Executar um comando SQL que insere um novo cliente na tabela.
 - Fazer o commit.
 - Fechar a conexão.

 Arquivo: `cliente_tela.py`


 Local: `src/telas/`

2. Crie a função privada `__cadastrar()`.
 - Solicite ao usuário o nome e o CPF usando a biblioteca `questionary`.
 - Chame a função `cadastrar` do repositório, passando os dados coletados.
 - Mostre uma mensagem de sucesso ao usuário.
3. Na função `executar_cliente()`, adicione a opção "Cadastrar" ao menu.
 - Quando o usuário selecionar essa opção, chame a função `__cadastrar()`.
4. No arquivo `main.py`, inclua no menu principal a opção "Clientes".
 - Quando selecionada, chame a função `executar_cliente()`.

Etapa 3 – Listar Todos os Clientes (SELECT)


 Arquivo: `cliente_repositorio.py`

1. Crie a função `listar_todos()` .
 - Essa função deve abrir a conexão com o banco e executar uma consulta para buscar todos os clientes.
 - Ela deve retornar uma lista de dicionários com os campos `id` , `nome` e `cpf` .


 Arquivo: `cliente_tela.py`

2. Crie a função privada `__listar_todos()` .
 - Chame a função `listar_todos()` do repositório.
 - Exiba os dados retornados de forma organizada no terminal.
 3. Na função `executar_cliente()` , adicione a opção "Listar todos".
 - Quando o usuário escolher essa opção, chame a função `__listar_todos()` .
-

Etapa 4 – Editar Cliente (UPDATE)

 Arquivo: `cliente_repositorio.py`

1. Crie a função `editar(id_editar: int, novo_nome: str, novo_cpf: str)` .
 - Ela deve abrir a conexão com o banco.
 - Executar um comando de atualização que altera o nome e o CPF com base no ID fornecido.
 - Fazer o commit e fechar a conexão.


 Arquivo: `cliente_tela.py`

2. Crie a função privada `__editar()` .
 - Solicite ao usuário o ID do cliente que deseja editar.
 - Peça também o novo nome e o novo CPF.
 - Chame a função `editar()` do repositório com esses dados.
 - Exiba uma mensagem confirmando a alteração.
 3. No menu da função `executar_cliente()` , adicione a opção "Editar".
 - Ao selecioná-la, chame a função `__editar()` .
-

Etapa 5 – Apagar Cliente (DELETE)

 Arquivo: `cliente_repositorio.py`

1. Crie a função `apagar(id_apagar: int)` .
 - Essa função deve abrir a conexão com o banco.
 - Executar um comando de exclusão com base no ID fornecido.
 - Fazer o commit e fechar a conexão.

 Arquivo: `cliente_tela.py`

2. Crie a função privada `__apagar()` .
 - Solicite ao usuário o ID do cliente a ser apagado.
 - Chame a função `apagar()` do repositório com esse ID.
 - Mostre uma mensagem de confirmação.
3. No menu da função `executar_cliente()` , adicione a opção "Apagar".
 - Ao selecioná-la, chame a função `__apagar()` .