Universidade Estácio de Sá

curso CIênCIAS DA COMPUTAÇÃO

UNIDADE NOVA AMÉRICA

**projeto rad**

**EM banco de dados**

**(Desenvolvimento rapido em python)**

Rio de Janeiro - RJ

Outubro / 2024

202304057214 – Gabriel Siqueira França

**Trabalho de projeto RAD**

**em (Desenvolvimento rápido em Python)**

Trabalho de Projeto Rad apresentado a Universidade Estácio de Sá, como exigência para avaliação na disciplina Desenvolvimento rápido em Python

Orientador:

Prof. Ronaldo Candido dos Santos

SUMÁRIO

[1 INTRODUÇÃO 3](#_Toc181707998)

[1.1 DESCRIÇÃO DO PROBLEMA 3](#_Toc181707999)

[1.2 OBJETIVOS 3](#_Toc181708000)

[2 DESENVOLVIMENTO 3](#_Toc181708001)

[2.1 Metodologia – RAD 3](#_Toc181708002)

[2.2 Modelagem de Dados 5](#_Toc181708003)

[2.3 Modelagem de Negócio 5](#_Toc181708004)

[2.4 Tecnologias Utilizadas 5](#_Toc181708005)

[2.5 Estrutura do Projeto 6](#_Toc181708006)

[2.6 Funcionalidades Principais 6](#_Toc181708007)

[3 CONCLUSÃO 7](#_Toc181708008)

[REFERÊNCIAS 9](#_Toc181708009)

# INTRODUÇÃO

Este projeto RAD (Rapid Application Development) implementa um sistema de gerenciamento de clientes, que permite cadastrar, consultar, alterar e excluir clientes em uma base de dados PostgreSQL, com uma interface gráfica amigável desenvolvida em Tkinter. A solução visa atender empresas de pequeno e médio porte que precisam gerenciar dados de clientes de forma simples e eficiente.

Através deste projeto, busca-se explorar a integração entre a linguagem Python e bancos de dados relacionais, além de aplicar conceitos de manipulação de dados, tratamento de exceções e operações CRUD (Create, Read, Update, Delete).

## DESCRIÇÃO DO PROBLEMA

Pequenas empresas geralmente enfrentam dificuldades ao gerenciar dados de clientes de forma manual, o que leva a processos lentos e sujeitos a erros. Um sistema informatizado que organiza as informações de clientes, facilitando a consulta e manutenção dos dados, é fundamental para otimizar a eficiência e reduzir erros operacionais.

## OBJETIVOS

O objetivo deste projeto é criar um sistema que ofereça uma interface intuitiva para que os usuários possam:

* **Cadastrar novos clientes** com informações de código, CPF, nome e idade.
* **Consultar e listar clientes** registrados.
* **Alterar dados de clientes** de maneira rápida e segura.
* **Excluir clientes** do sistema de forma segura, com confirmações para evitar exclusões acidentais.

# DESENVOLVIMENTO

O desenvolvimento do sistema foi realizado em Python, utilizando Tkinter para a interface gráfica e PostgreSQL para a base de dados. O projeto foi modularizado, distribuindo as funções entre quatro arquivos principais para facilitar a manutenção e evolução do sistema.

## 2.1 Metodologia – RAD

O método RAD foi aplicado em várias etapas de desenvolvimento, seguindo as fases clássicas de análise, design, construção e testes. Abaixo estão as etapas seguidas:

1. Planejamento e Definição de Requisitos:

* Nesta etapa, foram definidos os requisitos do sistema, como cadastro, consulta, alteração e exclusão de clientes.
* Foram identificados os principais usuários do sistema, suas necessidades e as interações desejadas.

1. **Modelagem de Negócio**:

* A modelagem de negócio foi realizada para definir o fluxo de operações e garantir que cada funcionalidade correspondesse a uma necessidade específica do usuário.
* A modelagem incluiu o fluxo para operações CRUD (Create, Read, Update, Delete) e as ações dos botões na interface gráfica.
* **Fluxo de Negócio**: O fluxo de negócio segue a lógica de um sistema de gerenciamento de clientes, onde o usuário realiza operações como cadastrar um cliente, visualizar clientes já cadastrados, realizar buscas e exclusões conforme necessário

1. **Modelagem de Dados**:

* A modelagem de dados foi realizada para estruturar as informações no banco de dados.
* O banco de dados PostgreSQL contém uma tabela principal chamada cadastro, que armazena os dados dos clientes.
* **Estrutura da Tabela:**
* **codigo**: Código único para cada cliente (chave primária).
* **cpf**: CPF do cliente.
* **nome**: Nome completo do cliente.
* **idade**: Idade do cliente.

1. **Construção Rápida**:

* O sistema foi dividido em módulos (main.py, interface.py, funcoes.py e classes.py) para facilitar a construção de forma rápida e segmentada.
* A cada protótipo, foram realizados testes e ajustes antes de avançar para a próxima funcionalidade.

1. **Testes e Validação**:

* Testes foram realizados para cada operação do sistema (inserção, consulta, atualização e exclusão).
* A interface gráfica foi validada para garantir que a experiência do usuário fosse intuitiva e funcional.

1. **Feedback e Ajustes:**

* Após cada iteração de desenvolvimento, os protótipos foram avaliados e, com base no feedback dos testes, melhorias foram implementadas até alcançar a versão final.

## 2.2 Modelagem de Dados

A modelagem de dados foi simplificada, utilizando uma única tabela (cadastro) para armazenar as informações de cada cliente.

**Diagrama da Tabela cadastro:**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| codigo | INT (PK) | Identificador único |
| cpf | VARCHAR(11) | CPF do cliente |
| nome | TEXT | Nome completo do cliente |
| idade | INT | Idade do cliente |

## 2.3 Modelagem de Negócio

A modelagem de negócio para este projeto foi baseada nos processos comuns de gerenciamento de clientes. O fluxo de trabalho foi organizado para atender a um fluxo sequencial e intuitivo para o usuário:

1. **Cadastro**: O usuário insere os dados do cliente (CPF, nome e idade) para registrá-lo no banco de dados.

* Para garantir a integridade do dado de CPF, foi implementada uma função limitar\_cpf para validar e restringir o número de caracteres. Essa função filtra a entrada para permitir apenas dígitos numéricos e limita o campo a 11 caracteres, correspondendo ao formato padrão de CPF.

1. **Consulta e Listagem**: Os dados dos clientes são exibidos em uma interface em formato de lista. Há uma opção de busca que permite ao usuário filtrar clientes por diferentes critérios.
2. **Alteração de Dados**: Para alterar os dados de um cliente, o usuário seleciona o cliente na lista e insere as novas informações. O sistema exibe uma mensagem de confirmação de sucesso.
3. **Exclusão de Clientes**: O usuário pode selecionar um cliente para exclusão. A exclusão requer uma confirmação para evitar a remoção acidental de dados.

## 2.4 Tecnologias Utilizadas

* **Python 3.x**: Linguagem de programação utilizada no projeto.
* **PostgreSQL**: Banco de dados relacional utilizado para armazenar os dados dos clientes.
* **Tkinter**: Biblioteca padrão do Python para construção de interfaces gráficas.
* **psycopg2**: Biblioteca para a conexão entre Python e PostgreSQL.

## 2.5 Estrutura do Projeto

├── main.py #Programa principal.

├── classes.py #Define a classe Cliente e outras classes necessárias.

├── funcoes.py #Contém funções para conexão com o banco e operações CRUD.

├── interface.py #Interface gráfica para interação do usuário.

└── README.md #Documentação do programa.

## 2.6 Funcionalidades Principais

**Arquivo: main.py**

**Este arquivo inicializa o sistema e executa as seguintes operações principais:**

1. **Criação da Tabela**: Chama a função criar\_tabela() em funcoes.py para garantir que a tabela cadastro exista no banco de dados.
2. **Inicialização da Interface**: Cria uma instância da classe **App** em **interface.py** para exibir a interface gráfica.

**Arquivo: funcoes.py**

O módulo funcoes.py é responsável por todas as operações relacionadas ao banco de dados, incluindo:

* **Conectar e Fechar Conexão**: Funções conectar\_banco() e fechar\_conexao() para gerenciar a conexão com o PostgreSQL.
* **Gerar Código**: A função gerar\_codigo() cria um novo código de cliente automaticamente, incrementando o valor máximo existente.
* **CRUD**:
  + criar\_tabela(): Cria a tabela **cadastro** com os campos **codigo**, **cpf**, **nome** e **idade**.
  + cadastrar\_cliente(): Insere um novo cliente no banco com **CPF**, **nome** e **idade**.
  + alterar\_dados(): Altera informações de um cliente específico, conforme os dados fornecidos (**CPF**, **nome**, **idade**).
  + deletar\_dados(): Exclui um cliente com base no **código**, após confirmação do usuário.
  + listar\_tabela(): Retorna e exibe todos os clientes cadastrados.

**Arquivo interface.py**

O módulo **interface.py** define a interface gráfica utilizando Tkinter e permite que o usuário interaja de forma intuitiva com o sistema:

1. **Tela de Cadastro**:
   * Permite que o usuário insira CPF, nome e idade para cadastrar novos clientes.
   * Exibe mensagens de sucesso ou erro ao finalizar o cadastro.
2. **Tela de Consulta**:
   * Utiliza um Treeview para exibir a lista de clientes cadastrados.
   * Inclui um campo de busca que permite filtrar clientes com base no nome, CPF ou código.
   * Exibe informações como código, CPF, nome e idade.
   * Permite ordenar a lista de clientes por qualquer coluna.
3. **Funções de Alteração e Exclusão**:
   * Permite alterar os dados de um cliente selecionado, exibindo um formulário de edição.
   * A exclusão de clientes requer confirmação e remove o cliente do banco caso confirmada.

**Arquivo classes.py**

Define a classe Cliente, que representa os clientes no sistema e organiza os atributos **codigo**, **cpf**, **nome** e **idade**.

# CONCLUSÃO

O sistema de gerenciamento de clientes criado cumpre seu objetivo de simplificar o armazenamento e a manipulação de dados de clientes em um ambiente corporativo. A utilização do PostgreSQL garante a integridade dos dados e permite uma fácil manutenção, enquanto o Tkinter proporciona uma interface amigável e intuitiva para usuários finais.

O projeto pode ser ampliado para incluir novas funcionalidades, como:

* **Relatórios**: Geração de relatórios em PDF ou CSV.
* **Filtros de Busca**: Filtros de busca mais avançados
* **Interface Web**: Migração da interface gráfica para uma plataforma web para acessibilidade remota.

# REFERÊNCIAS

ESTÁCIO. Conteúdo digital da disciplina, Temas “Rad (rapid Applications Development)”, “Manipulação de Dados em Arquivos”, “Python Com Banco de Dados”, "Interface Gráfica com Python", e “Aplicando Rad

Aplican” Módulos 2, 3 , 4, 5 , 6 . Disponível na Sala de aula virtual da disciplina.

<https://estudante.estacio.br/disciplinas/estacio\_12976930/temas/>

NORMAS ABNT. Normas para elaboração de trabalhos acadêmicos. Disponível em: <https://www.normasabnt.org/>. Acesso em: 28 mai. 2024.

CHATGPT. Disponível em:

<https://chatgpt.com/>