**UNIVERSIDADE ESTÁCIO DE SÁ**

JEFFERSON PEDRO

THIAGO COROA

**EXERCÍCIO AVALIATIVO AV**

**IMC CALCULAPRO**

Projeto complementar para nota final da disciplina de Desenvolvimento Rápido em Python.

Professor(a): Prof. Simone Gama

NITERÓ

2023

**IMC CalculaPro**

Numa cidade movimentada, a empresa de bem-estar "Viva Bem" percebeu a necessidade de proporcionar uma ferramenta mais acessível e interativa para seus clientes acompanharem sua saúde física. A empresa, especializada em programas de fitness e alimentação saudável, decidiu criar um aplicativo inovador chamado " **IMC CalculaPro** ".

A motivação por trás desse projeto surgiu após uma pesquisa interna revelar que muitos clientes tinham dificuldades em calcular e entender o Índice de Massa Corporal (IMC), uma medida essencial para avaliar a saúde física. A ideia era desenvolver uma solução que tornasse o monitoramento do IMC fácil e intuitivo.

**1 - A aplicação deverá ter:**

* 1. – Banco de dados;
  2. – Gerenciamento de usuários;
  3. – Tela de login;
  4. – Tela de Cadastro;
  5. – Cálculo do IMC;
  6. – Resultado do IMC;

**1.1 – Gerenciamento de Usuários:**

* Esta janela deverá conter as informações do usuário cadastrado. Tais como, a id, nome, email e username;
* Ter pelo menos três botões com opções para que o Admin possa limpar, excluir ou atualizar os dados do usuário;
* Ter campos para preencher e atualizar as informações dos usuários;
* Ter campos para preencher e excluir as informações dos usuários;
* Deverá conter uma aba de “ações”. Para que o Admin possa calcular o IMC, cadastrar um novo usuário e para que consiga sair da tela de gerenciamento.

**1.2 – Tela de Login:**

* Ter campo para preenchimento do login e senha de usuários e administrador;
* Ter dois botões, um para login e outro para cadastro de novos usuários;
* Caso o usuário insira o login ou a senha incorreta, ter pelo menos uma janela para que o certifique do ocorrido;
* Ter uma janela para que certifique o usuário de que o login foi bem sucedido;

**1.3 – Tela de Cadastro**:

* Deverá haver campos para que o novo usuário preencha os seus dados. Tais como, nome, email, telefone, usarname e senha;
* Deverá ter dois botões. Um para finalizar o cadastro e outro para limpar as informações preenchidas;

**1.4 – Cálculo do IMC:**

* Deverá haver campos para que o usuário cadastrado preencha com algumas informações. Tais como, o seu peso, altura e idade;
* Deverá haver uma opção para que o usuário escolha o seu gênero;
* Deverá haver um botão para o cálculo do IMC;
* Gerar uma mensagem correspondendo o resultado do IMC do usuário;

**2- Banco de dados:**

O banco de dados usado neste projeto é o PostgreSQL v15.1. Que possui uma tabela chamada tb\_users, dentro do schema exercicio\_av. Esta tabela possui 6 colunas: id, nome, email, telefone, username, senha.

Sendo, o ID do tipo “serial” auto incrementável e os demais campos como varying com espaço de até 50 caracteres.

Este repositório também possui um arquivo de texto chamado registro.txt que possui 20 usuários como exemplos a serem inseridos no banco para realização de testes mais o usuário admin.

**3- Arquivos do projeto:**

* 1. **main.py**: Esta classe é responsável pelo start do programa, instanciando a classe Login(), descrita no item 3.2;
  2. **GUI\_Login.py:** Esta classe é a responsável por toda a logica de login e autenticação da aplicação. O script cria uma interface de login simples com opções para cadastro, cálculo de IMC e uma interface de administração, usando a biblioteca Tkinter e interagindo com um banco de dados por meio do módulo personalizado bancoDeDados.  
      A classe **Login**, em seu **método construtor** é a responsável por inicializar os componentes widgets (labels, entrys, botões) e posicionar cada um na janela principal do Login.

O método **fazer\_login**, obtém as informações do usuário e senha, verifica as credenciais usando o método **verificar\_credenciais** do objeto **BancoDeDados** e exibe mensagens de sucesso ou erro.

O método **abrir\_cadastro, f**echa a janela de login, cria uma nova janela para o cadastro e a torna ativa (grab\_set()).

O método **abrir\_calculo\_imc** e **abrir\_interface\_admin**: Realizam ações semelhantes às descritas em **abrir\_cadastro**. Caso o usuário autenticado seja administrador, irá abrir a classe **abrir\_interface\_admin**. Caso seja usuário comum **abrir\_calculo\_imc.**

Por fim, temos ao final do script a inicialização da Janela Principal, onde:

- É criada uma instância da classe Tk para representar a janela principal. O título da janela é definido como "Tela de Login" e uma imagem é carregada e exibida na janela.

- O tamanho da janela é definido como 330x330 pixels, e a janela é configurada como não redimensionável.

- Uma instância da classe Login é criada, associada à janela principal (root) e o método mainloop() é chamado para iniciar o loop principal da aplicação, que aguarda eventos do usuário.

Obs: **Para logar como administrador use o login: admin e a senha: 123**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* 1. **GUI\_Cadastro.py:** Esta classe é responsável para criar uma interface gráfica de usuário (GUI) para o cadastro de usuários. O script cria uma interface gráfica para o cadastro de usuários usando Tkinter, onde os dados inseridos são enviados para um banco de dados por meio de métodos definidos no objeto BancoDeDados.

A classe **Cadastro** em seu método **construtor** é a responsável por inicializar os componentes widgets (labels, entrys, botões) e posicionar cada um na janela Cadastro.

O método **fCadastrar**, obtém os dados dos campos de entrada, chama o método **inserirDados** do objeto BancoDeDados para inserir os dados no banco e exibe mensagens de sucesso ou erro.

O método **fLimparTela**, limpa os campos de entrada na interface.

Por fim, temos ao final do script a inicialização da Janela Principal, onde:

- É criada uma instância da classe Tk para representar a janela principal.

- Um título é definido para a janela.

- Uma imagem é carregada e exibida na janela.

- Uma instância da classe Cadastro é criada e associada à janela principal.

- O tamanho da janela é definido como 400x500 pixels, e a janela é configurada como não redimensionável.

Interface gráfica do usuário, Aplicativo, Site

Descrição gerada automaticamente

3.4 **GUI\_Gerenciar.py:** Esta classe é responsável para gerenciar usuários. Este script é uma aplicação de gerenciamento de usuários com funcionalidades de atualização, exclusão e visualização de usuários, além de ações no menu para cadastrar novos usuários e calcular o IMC.

A classe **Gerenciar** em seu método **construtor** é a responsável por inicializar os componentes widgets (labels, entrys, botões) e posicionar cada um na janela Gerenciar.

O método **fListaUsuarios,** atualiza a lista de usuários exibida na Treeview (listaUsuários) a partir dos dados obtidos do banco de dados.

O método **fExcluiUsuario**, exclui um usuário com base no ID fornecido.

O método **fAtualizar**, atualiza os dados de um usuário com base no ID fornecido.

O método **fLimparTela**, limpa os campos de entrada na interface.

O método **fSelecionaUsuario**, preenche os campos de entrada com os dados do usuário selecionado na Treeview.

O método **configurar**\_**menu**, configura um menu na parte superior da janela para ações como calcular IMC, cadastrar novo usuário e sair.

O método **abrir**\_**cadastro** e **abrir\_calculo\_imc**, fecha a janela atual e abre uma nova janela para cadastro ou cálculo de IMC.

Por fim, temos ao final do script a inicialização da Janela Principal, onde:

- É criada uma instância da classe Tk.

- Configura a interface com dois frames (frame1 e frame2) para organizar os componentes.

- Usa a Treeview para exibir os usuários em forma de tabela.

- Define colunas e cabeçalhos para a Treeview.

- Configura uma barra de rolagem para a Treeview.

- Cria uma instância da classe Gerenciar associada à janela principal.

- Chamadas de Métodos Iniciais: **fListaUsuario, p**reenche a Treeview com dados do banco de dados para exibição inicial. E o mainloop(): Inicia o loop

Interface gráfica do usuário

Descrição gerada automaticamenteprincipal do Tkinter para aguardar eventos do usuário.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* 1. **GUI\_CalculoIMC.py:** Esta classe é responsável criar uma interface gráfica de usuário (GUI) para calcular o Índice de Massa Corporal (IMC). A interface permite que o usuário insira informações como peso, altura, gênero e idade, e ao pressionar o botão "Calcular IMC", o script realiza o cálculo e exibe uma mensagem com o resultado.

A classe **IMC** em seu método **construtor** é a responsável por inicializar os componentes widgets (labels, entrys, botões) e posicionar cada um na janela IMC.

O método **calcular\_imc**, obtém os valores de peso, altura, gênero e idade da interface, realiza o cálculo do IMC e exibe uma mensagem de informação com o resultado do cálculo e a classificação do IMC.

O método **classificar\_imc**, classifica o IMC em categorias como "Baixo peso", "Peso normal", "Sobrepeso", etc.

Por fim, temos ao final do script a inicialização da janela Principal, onde:   
- Cria uma instância da classe Tk para representar a janela principal.

- Define o título da janela como "Cálculo de IMC".

- Carrega uma imagem e a exibe na janela.

- Configura o tamanho da janela e a torna não redimensionável.

- Cria uma instância da classe IMC associada à janela principal.

- Inicia o loop principal do Tkinter para aguardar eventos do usuário.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* 1. **bancoDeDados.py**: Esta classe é responsável por realizar os métodos CRUD da aplicação e a verificação de usuários.

O método **construtor**, chama o método conexao para estabelecer a conexão com o banco de dados.

O método **conexão**, estabelece uma conexão com o banco de dados PostgreSQL usando as credenciais fornecidas, cria uma tabela chamada tb\_users se ela não existir, fecha a conexão após a execução.

O método **inserirDados**, insere dados (nome, email, telefone, username, senha) na tabela tb\_users, utiliza placeholders %s para evitar injeção de SQL, fecha a conexão após a execução.

O método **lerDados**, executa uma consulta SQL para obter todos os registros da tabela **tb\_users**, exibe os resultados no console, retorna os dados obtidos da consulta, fecha a conexão após a execução.

O método **atualizarDados**, atualiza dados na tabela **tb\_users** com base no ID fornecido, utiliza placeholders %s para evitar injeção de SQL, fecha a conexão após a execução.

O método **deletarDados**, deleta um registro na tabela **tb\_users** com base no ID fornecido, utiliza placeholders %s para evitar injeção de SQL, fecha a conexão após a execução.

O método **verificar\_credenciais, v**erifica as credenciais de um usuário na tabela **tb\_users**, utiliza placeholders %s para evitar injeção de SQL, retorna o usuário encontrado ou None se não encontrado, fecha a conexão após a execução.

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

**4- Link do Repositório:**

[Atividades-com-Python/Trabalhos/Exercicio\_Avaliativo\_AV at main · Jefferson-Pedro/Atividades-com-Python (github.com)](https://github.com/Jefferson-Pedro/Atividades-com-Python/tree/main/Trabalhos/Exercicio_Avaliativo_AV)