Fernando Lima Alves
Luciana - Python Intermediário
Curso - Qualificação Programação de computadores

Projeto Sistema de inscrição pessoal

Interface:

Primeiramente comecei criando um arquivo Python para poder estar iniciando o projeto. Iniciei com o nome de “Proeto.py” e criei o arquivo.

```
import tkinter as tk
from tkinter import messagebox

usuarios = []

#Criação da Janela Principal
janela = tk.Tk()
janela.title("Sistema De dados ")
janela.geometry("300x300")
titulo = tk.Label(janela, text="Sistema de inscrição")
```

Logo após estar criando, importei a biblioteca que será utilizada para a interface gráfica, onde já logo criei uma janela, assim atribuindo o nome da biblioteca como TK e assim facilitando a escrita. Logo abaixo, já pensando que o sistema seria para usuários, criei uma lista para receber todos os dados mais a frente.

Começando a programação, comecei criando uma janela onde iria conter todas os elementos gráficos, assim manipulei a janela com o título e o tamanho dela, assim definindo com 400 x 300 Pixels.

```
#Entrada de nome
tk.Label(janela, text="Nome").grid(row=0, column=0)#texto
entrar_nome = tk.Entry(janela)
entrar_nome.grid(row=0, column=1)#espaço para digitar

#Entrada de Idade

tk.Label(janela, text="Idade").grid(row=1, column=0)#texto
entrar_idade = tk.Entry(janela)
entrar_idade.grid(row=1, column=1)#espaço para digitar

#Entrada de Cpf

tk.Label(janela, text="CPF").grid(row=2, column=0)#texto
entrar_cpf = tk.Entry(janela)
entrar_cpf.grid(row=2, column=1)#espaço para digitar

#Entrada de Endereço
tk.Label(janela, text="Enderço").grid(row=3, column=0)#texto
entrar_endereco = tk.Entry(janela)
entrar_endereco.grid(row=3, column=1) #espaço para digitar

#Entrada de Profição
tk.Label(janela, text="Profissao").grid(row=4, column=0) #texto
entrar_profissao = tk.Entry(janela)
entrar_profissao.grid(row=4, column=1) #espaço para digitar
```

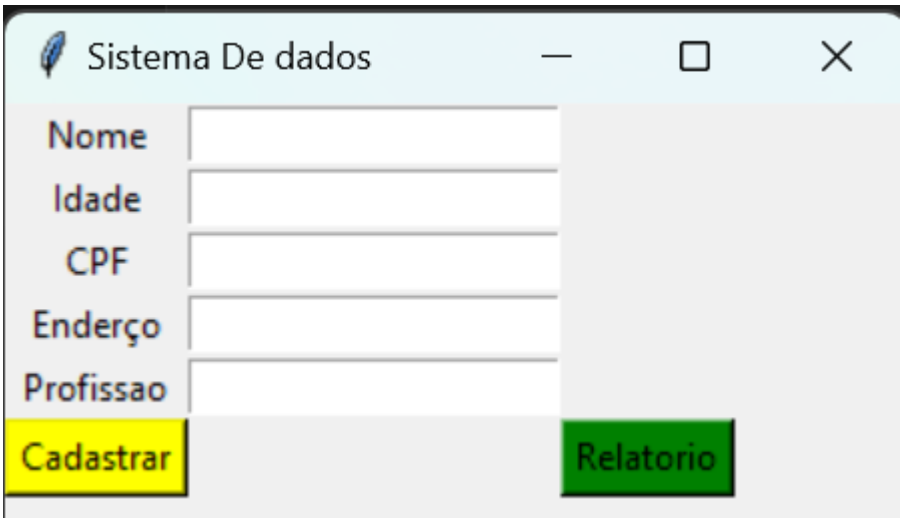
Logo após, criar espaços para poder estar entrando e utilizando a tag os dados junto com um espaço para dizer o'que é para digitar, assim criando um para cada um sendo representado como:

Assim ficando o com o espaço dizendo para o usuário o que é para digitar, assim tendo o espaço para poder estar colocando seus dados.

```
btn_cadastrar = tk.Button(j (parameter) column: int = "yellow", command = cadastrar_usuario)
btn_cadastrar.grid(row=10, column=0)

#Botão Relatorio
btn_relatorio  = tk.Button(janela, text="Relatorio", bg="green", command= relatorio)
btn_relatorio.grid(row=10, column=3)
```

Logo após, utilizei da função que já está configurada no Tkinter, que é o botão, e assim criei dois botões com funções utilizando a tag **command**, para serem realizadas quando houver algum clique. Posicionei eles em lugares para melhor visualização utilizando a função grid do Tkinter:



Fiz mudanças de cor para ter uma facilidade para enxergar e ter uma identidade visual para o usuário entender o que está sendo feito, sem ler várias vezes o botão.

Cadastro :

```python
def cadastrar_usuario():
    nome = entrar_nome.get()
    idade = entrar_idade.get()
    cpf = entrar_cpf.get()
    endereco = entrar_endereco.get()
    profissao = entrar_profissao.get()

    # Verificar campos vazios
    if not nome or not idade or not cpf or not endereco or not profissao:
        messagebox.showerror("Erro", "Todos os campos devem ser preenchidos.")
        return

    # Validações específicas
    if any(char.isdigit() for char in nome):
        messagebox.showerror("Erro", "O nome não pode conter números.")
        return

    if any(char.isalpha() for char in idade):
        messagebox.showerror("Erro", "A idade não pode conter letras.")
        return

    if any(char.isalpha() for char in cpf):
        messagebox.showerror("Erro", "O CPF não pode conter letras.")
        return

    if any(char.isdigit() for char in profissao):
        messagebox.showerror("Erro", "A profissão não pode conter números.")
        return

    # Cadastro válido
    usuario = {
        'nome': nome,
        'idade': idade,
        'cpf': cpf,
        'endereco': endereco,
        'profissao': profissao
    }

    usuarios.append(usuario)
    messagebox.showinfo("Cadastro concluído", "Seu cadastro foi efetuado com sucesso!")

    # Limpar os campos após cadastro
    entrar_nome.delete(0, tk.END)
    entrar_idade.delete(0, tk.END)
    entrar_cpf.delete(0, tk.END)
    entrar_endereco.delete(0, tk.END)
    entrar_profissao.delete(0, tk.END)
```

Para fazer o cadastro do usuário, primeiramente criamos uma função que se chama **Cadastrar_usuario**, onde ela vai receber todos os os seguintes funcionalidades (Irá receber os dados dos campos, irá verificar se são compatíveis, com o'que está pedindo e se forem irá ser adicionado ao um dicionário que esse dicionário irá receber em uma lista já

criada no começo). Utilizamos a função **.get()**, para poder pegar todos os dados daquele campo que foi direcionado assim retornando o'que foi escrito em uma String, mas dependendo de qual foi o dado utilizado precisamos de um Inteiro ou um Float, assim utilizamos uma função chamada **any()**, onde ela fazer qualquer coisa que você pedir, onde foi utilizado outra função **char** que verifica os caracteres escritos e verifica se é um dígito com outra função **isdigit()** que fazemos rodar em nosso nome escrito ou número, assim fazer um **for** para rodar em cada letra do que foi digitado assim alterando de **isdigit()** **isalpha()** para verificar se tem algum letra em algum número.

```
# Validações específicas
if any(char.isdigit() for char in nome):
    messagebox.showerror("Erro", "O nome não pode conter números.")
```

Nesse exemplo verifica se tem um número na variável nome com o processo dito acima. Caso tenha, ele retorna um erro para o usuário

```
if any(char.isalpha() for char in idade):
    messagebox.showerror("Erro", "A idade não pode conter letras.")
    return
```

Neste outro exemplo verifica se tem alguma letra na variável idade.Caso tenha, ele retorna um erro para o usuário.

```
# Verificar campos vazios
if not nome or not idade or not cpf or not endereco or not profissao:
    messagebox.showerror("Erro", "Todos os campos devem ser preenchidos.")
    return
```

Neste campo ele irá verificar se em todas as variáveis tenha algo escrito, caso alguma ou todas não tenha nada digitado ele irá passar a mensagem de erro para o usuário.

```python
# Cadastro válido
usuario = {
    'nome': nome,
    'idade': idade,
    'cpf': cpf,
    'endereco': endereco,
    'profissao': profissao
}

usuarios.append(usuario)
messagebox.showinfo("Cadastro concluído", "Seu cadastro foi efetuado com sucesso!")

# Limpar os campos após cadastro
entrar_nome.delete(0, tk.END)
entrar_idade.delete(0, tk.END)
entrar_cpf.delete(0, tk.END)
entrar_endereco.delete(0, tk.END)
entrar_profissao.delete(0, tk.END)
```

Caso tudo aquilo lá em cima dê certo, ele irá adicionar essa “Pessoa” no dicionário pessoa, atribuindo cada valor de cada variável, assim adicionando a pessoa que foi criado agora, na lista de “Pessoas”.

```python
# Limpar os campos após cadastro
entrar_nome.delete(0, tk.END)
entrar_idade.delete(0, tk.END)
entrar_cpf.delete(0, tk.END)
entrar_endereco.delete(0, tk.END)
entrar_profissao.delete(0, tk.END)
```

Foi utilizado a função **delete** para poder estar limpando os campos assim ficando mais fácil para o usuário escrever após apertar o botão, assim ficando melhor a experiência do usuário.

Relatório:

```python
# Botão Relatório
def relatorio():
    if not usuarios:
        messagebox.showinfo("Relatório", "Nenhum usuário cadastrado ainda.")
        return

    relatorio = ""
    for u in usuarios:
        linha = (
            f"{u['nome']} tem {u['idade']} anos, CPF: {u['cpf']}, "
            f"mora em {u['endereco']} e trabalha como {u['profissao']}.\n\n"
        )
        relatorio += linha

    messagebox.showinfo("Relatório de Usuários", relatorio)
```

Por fim, criamos outra função chamada relatório, atribuída ao outro botão criado anteriormente, onde primeiramente ele verifica se existe algum usuário criado, e caso não tenha ele emite uma mensagem de erro, dizendo que não tem nenhum usuário cadastrado ainda.
Caso já tenha algum usuário, cria se uma variável “**Relatório**” onde ela recebe uma string vazia para que ela comece a criar o relatório, que verifica usuário por usuário na lista de usuários e pega de cada usuário, seu nome, sua idade , seu cpf, seu endereço, e sua profissão e adiciona em uma variável chamada linha. que logo após a chamada e adicionada no relatório, que irá ser exibido no final para o usuário, contendo tudo que foi inscrito no cadastro