

# INTERFACE GRÁFICA

O Python é uma linguagem de scripts utilizado principalmente para automação ou análise de dados. Ou também integração com algum framework ou api atuando como backend de um software. Porém conseguimos utilizar algumas bibliotecas de UI. Criando a interface de Janelas de uma programa.

Vamos utilizar 3 bibliotecas de UI do Python o PYQT5 Tkinter e o PySimpleGUI

Vamos começar utilizando o PYQT5

Vamos instalar as bibliotecas do PyQT5

```
pip install pyqt5
```

```
pip install pyqt5-tools
```

# INTERFACE GRÁFICA

Criando Primeiro Programa com PyQt5.

```
from PyQt5.QtWidgets import *
```

```
app = QApplication([])  
label = QLabel('Hello World!')  
label.show()  
app.exec_()
```

Criando programas com botões e widgets.

```
from PyQt5.QtWidgets import *  
  
app = QApplication([])  
button = QPushButton('Botão Clique')
```

# INTERFACE GRÁFICA

Função do Botao

```
def on_button_clicked():  
    alert = QMessageBox()  
    alert.setText('Mensagem Gerada com Sucesso!')  
    alert.exec_()
```

Ação Botao

```
button.clicked.connect(on_button_clicked)
```

```
button.show()  
app.exec()
```

# INTERFACE GRÁFICA

Integrando uma interface gráfica criada pelo QtDesigner. Vamos utilizar o arquivo salvo do QtDesigner main.ui

```
from PyQt5 import uic  
from PyQt5.QtWidgets import QApplication
```

```
Form, Window = uic.loadUiType("main.ui")
```

```
app = QApplication([])  
window = Window()  
form = Form()  
form.setupUi(window)  
window.show()  
app.exec()
```

# INTERFACE GRÁFICA

Convertendo Arquivos .ui criados no QTDesigner em arquivos de script Python.

Vamos converter os arquivos main.ui e confirma.ui em scripts Python.

Vamos utilizar os comandos abaixo para converter os arquivos utilizando o pyuic5:

```
pyuic5 -x confirma.ui -o confirma.py  
pyuic5 -x main.ui -o main.py
```

# INTERFACE GRÁFICA

Vamos integrar as 2 janelas do main.py com o confirma.py

Vamos adicionar os seguintes códigos no arquivo main.py

Importar a classe no arquivo principal

```
from confirma import Ui_MainWindow
```

Criar a classe de instancia do Objeto de Abrir uma Janela

```
def openWindow(self):  
    self.window = QtWidgets.QMainWindow()  
    self.ui = Ui_MainWindow()  
    self.ui.setupUi(self.window)  
    self.window.show()
```



# INTERFACE GRÁFICA

Vincular o clique com a classe de abrir a janela.

```
self.Cadastrar.clicked.connect(self.openWindow)
```

Vamos utilizar outra biblioteca de ui padrão do Python o TKINTER. Vamos criar um novo arquivo e instanciar um janela.

```
.  
import tkinter as tk
```

Cria a Janela Principal

```
window = tk.Tk()
```

```
saudacao = tk.Label(text="Olá, Python UI")
```

```
saudacao.pack()
```

Executar o Tkinter

```
window.mainloop()
```

# INTERFACE GRÁFICA

Criando uma janela completa completa com Tkinter com widgets e botões.

```
import tkinter as tk
```

```
class tela(tk.Frame):
```

```
    def __init__(self, master=None):
```

```
        super().__init__(master)
```

```
        self.master = master
```

```
        self.pack()
```

```
        self.cria_widgets()
```

```
    def cria_widgets(self):
```

```
        self.acao_botao = tk.Button(self)
```

```
        self.acao_botao["text"] = "Olá\n(Clique Aqui)"
```

```
        self.acao_botao["command"] = self.diz_ola
```

```
        self.acao_botao.pack(side="top")
```



# INTERFACE GRÁFICA

```
def cria_widgets(self):  
    self.acao_botao = tk.Button(self)  
    self.acao_botao["text"] = "Olá\n(Clique Aqui)"  
    self.acao_botao["command"] = self.diz_ola  
    self.acao_botao.pack(side="top")  
  
    self.sair = tk.Button(self, text="Sair", fg="red",  
        command=self.master.destroy)  
    self.sair.pack(side="bottom")  
  
def diz_ola(self):  
    print("Olá Testando Saida em Console")  
  
window = tk.Tk()  
app = tela(master=window)  
app.mainloop()
```

# INTERFACE GRÁFICA

Criando um programa de cadastro de Usuários integrado ao banco de dados MYSQL utilizado nas aulas anteriores.

Primeiramente vamos criar os programas com as classes de conexão. Inserção e busca no banco de dados e seus atributos.

Utilizaremos classes para efetuar a comunicação com o banco de dados.

```
import pymysql
```

```
conn = pymysql.connect(db='cadastro_clientes', user='user', passwd='Python@123')
```

```
class Tabela_cadastro(object):
```

```
    def __init__(self, id= 0 ,nome="", sobrenome="",cpf=""):
        self.id = id
        self.nome = nome
        self.sobrenome = sobrenome
        self.cpf = cpf
```

# INTERFACE GRÁFICA

```
def insere_user(self):  
  
    cursor = conn.cursor()  
    cursor.execute("insert into cadastro_clientes (nome, sobrenome, cpf) values ('" + self.nome  
+ "', '" + self.sobrenome + "', '" + self.cpf + "')")  
    conn.commit()  
    cursor.close()
```

# INTERFACE GRÁFICA

```
def selectusuario(self, id):  
  
    cursor = conn.cursor()  
  
    cursor.execute("select * from cadastro_clientes where id = " + id + " ")  
  
    for regs in cursor:  
        self.id = regs[0]  
        self.nome = regs[1]  
        self.sobrenome = regs[2]  
        self.cpf = regs[3]  
  
    cursor.close()
```

# INTERFACE GRÁFICA

Criando o Programa com a Interface Grafica utilizando containers ao invés de widgets e recursos de labels e botões do Tkinter.

Importando os módulos e classes.

```
from pytkuser import Tabela_cadastro  
from tkinter import *
```

```
class Cadastro_Cliente:  
    def __init__(self, master=None):
```

```
        self.fonte = ("Verdana", "8")
```

```
        self.container1 = Frame(master)  
        self.container1["pady"] = 10  
        self.container1.pack()
```



# INTERFACE GRÁFICA

```
self.container2 = Frame(master)
self.container2["padx"] = 20
self.container2["pady"] = 5
self.container2.pack()
self.container3 = Frame(master)
self.container3["padx"] = 20
self.container3["pady"] = 5
self.container3.pack()
self.container4 = Frame(master)
self.container4["padx"] = 20
self.container4["pady"] = 5
self.container4.pack()
self.container5 = Frame(master)
self.container5["padx"] = 20
self.container5["pady"] = 5
self.container5.pack()
```



# INTERFACE GRÁFICA

```
self.container6 = Frame(master)
self.container6["padx"] = 20
self.container6["pady"] = 5
self.container6.pack()
self.container7 = Frame(master)
self.container7["padx"] = 20
self.container7["pady"] = 5
self.container7.pack()

self.titulo = Label(self.container1, text="Informe o Cadastro :")
self.titulo["font"] = ("Calibri", "12", "bold")
self.titulo.pack ()

self.lbidcadastro = Label(self.container2,
text="idCadastro:", font=self.fonte, width=10)
self.lbidcadastro.pack(side=LEFT)
```

# INTERFACE GRÁFICA

```
self.txtidcadastro = Entry(self.container2)
self.txtidcadastro["width"] = 10
self.txtidcadastro["font"] = self.fonte
self.txtidcadastro.pack(side=LEFT)
```

```
self.btnBuscar = Button(self.container2, text="Buscar",
font=self.fonte, width=10)
self.btnBuscar["command"] = self.buscarcadastro
self.btnBuscar.pack(side=RIGHT)
```

```
self.lblnome = Label(self.container3, text="Nome:",
font=self.fonte, width=10)
self.lblnome.pack(side=LEFT)
```

```
self.txtnome = Entry(self.container3)
self.txtnome["width"] = 25
self.txtnome["font"] = self.fonte
self.txtnome.pack(side=LEFT)
```

# INTERFACE GRÁFICA

```
self.lblsobrenome = Label(self.container4, text="Sobrenome:",  
font=self.fonte, width=10)  
self.lblsobrenome.pack(side=LEFT)
```

```
self.txtsobrenome = Entry(self.container4)  
self.txtsobrenome["width"] = 25  
self.txtsobrenome["font"] = self.fonte  
self.txtsobrenome.pack(side=LEFT)
```

```
self.lblcpf= Label(self.container5, text="CPF:",  
font=self.fonte, width=10)  
self.lblcpf.pack(side=LEFT)
```

```
self.txtcpf = Entry(self.container5)  
self.txtcpf["width"] = 25  
self.txtcpf["font"] = self.fonte  
self.txtcpf.pack(side=LEFT)
```

# INTERFACE GRÁFICA

```
self.bntInsert = Button(self.container6, text="Inserir",  
font=self.fonte, width=12)  
self.bntInsert["command"] = self.inserirUsuario  
self.bntInsert.pack (side=LEFT)
```

```
self.lblmsg = Label(self.container7, text="")  
self.lblmsg["font"] = ("Verdana", "9", "italic")  
self.lblmsg.pack()
```

# INTERFACE GRÁFICA

```
def inserirUsuario(self):  
    user = Tabela_cadastro()  
  
    user.nome = self.txtnome.get()  
    user.sobrenome = self.txtsobrenome.get()  
    user.cpf = self.txtcpf.get()  
  
    self.lblmsg["text"] = user.insere_user()  
  
    self.txtidcadastro.delete(0, END)  
    self.txtnome.delete(0, END)  
    self.txtsobrenome.delete(0, END)  
    self.txtcpf.delete(0, END)
```



# INTERFACE GRÁFICA

```
def buscarcadastro(self):  
    user = Tabela_cadastro()  
  
    id = self.txtidcadastro.get()  
  
    self.lblmsg["text"] = user.selectusuario(id)  
  
    self.txtidcadastro.delete(0, END)  
    self.txtidcadastro.insert(INSERT, user.id)  
  
    self.txtnome.delete(0, END)  
    self.txtnome.insert(INSERT, user.nome)  
  
    self.txtsobrenome.delete(0, END)  
    self.txtsobrenome.insert(INSERT, user.sobrenome)  
  
    self.txtcpf.delete(0, END)  
    self.txtcpf.insert(INSERT, user.cpf)
```



# INTERFACE GRÁFICA

```
root = Tk()  
Cadastro_Cliente(root)  
root.mainloop()
```

# INTERFACE GRÁFICA

Trabalhando com a Biblioteca PySimpleGUI

Primeiro vamos instalar a biblioteca PySimpleGUI

```
pip install pysimplegui
```

Criando uma janela simples.

```
import PySimpleGUI as sg
```

```
layout = [[sg.Text("Ola Estou Criando uma Mensagem")], [sg.Button("OK")]]
```

# INTERFACE GRÁFICA

# Criando a Janela

```
window = sg.Window("TITULO", layout)
```

# Criando o loop do evento

```
while True:
```

```
    event, values = window.read()
```

```
    # Finalize o programa se o usuário fechar a janela ou pressiona OK
```

```
    if event == "OK" or event == sg.WIN_CLOSED:
```

```
        break
```

```
window.close()
```

# INTERFACE GRÁFICA

Criando outra interface com Saída em Janela Gráfica

```
import PySimpleGUI as sg
```

```
# Definindo Conteúdo da Janela
```

```
layout = [[sg.Text("Qual Seu Nome ?")],  
          [sg.Input(key='-INPUT-')],  
          [sg.Text(size=(40,1), key='-OUTPUT-')],  
          [sg.Button('Ok'), sg.Button('Sair')]]
```

```
# Criando a Janela
```

```
window = sg.Window('Titulo da Janela', layout)
```

# INTERFACE GRÁFICA

# Exibir e interagir com a janela usando um loop de eventos

```
while True:
```

```
    event, values = window.read()
```

```
    # Validação usuario sair ou a janela foi fechada
```

```
    if event == sg.WINDOW_CLOSED or event == 'Sair':
```

```
        break
```

```
    # Criando a saída da Mensagem na Interface
```

```
    window['-OUTPUT-'].update('Olá ' + values['-INPUT-'] + " Gravando Saida na Interface")
```

```
window.close()
```

# INTERFACE GRÁFICA

Por ultimo vamos rodar um programa visualizador de imagens criados em PySimpleGUI o mesmo irá buscar arquivos de imagem no computador em formato png ou gif e exibir na interface do Python.

Vamos abrir o arquivo `visualizador_img.py` anexado ao material do curso.



# INTERFACE GRÁFICA

## REFERENCIAS:

<https://github.com/pyqt/examples/commits?author=mherrmann>

<https://www.youtube.com/watch?v=dRRpbDFnMHI>

<https://docs.python.org/3/library/tkinter.html>

[https://www.python-course.eu/tkinter\\_buttons.php](https://www.python-course.eu/tkinter_buttons.php)

<https://realpython.com/python-gui-tkinter/>

<https://realpython.com/pysimplegui-python/>

<https://pypi.org/project/PySimpleGUI/>