

# USANDO PYTHON

## VOLUME - IV



**AVAILABLE IN**

## PREPARADO POR

*Rajyalaxmi Singam*  
*João Futi Muanda*

# USANDO PYTHON VOLUME - IV

P O R

RAJYALAXMI SINGAM  
&  
JOÃO FUTU MUANDA

Copyright © 2021 by Usando Python

TODOS OS DIREITOS RESERVADOS. NENHUMA PARTE DESTA PUBLICAÇÃO PODE SER REPRODUZIDA, DISTRIBUÍDA OU TRANSMITIDA DE QUALQUER FORMA OU POR QUALQUER MEIO, INCLUINDO FOTOCÓPIA, GRAVAÇÃO OU OUTROS MÉTODOS ELETRÔNICOS OU MECÂNICOS, SEM A PERMISSÃO PRÉVIA POR ESCRITO DO EDITOR, EXCETO NO CASO DE BREVES CITAÇÕES INCORPORADAS EM REVISÕES CRÍTICAS E OUTROS USOS NÃO COMERCIAIS PERMITIDOS PELA LEI DE DIREITOS AUTORAIS.

# Para quem é este Guia?

Este guia é para pessoas com conhecimento básico de Python, bem como para aqueles que se sentem um pouco perdidos e não sabem como usar Python na prática. E é também para quem quer praticar Python de uma forma um pouco diferente que fará com que você saiba o que está fazendo e não apenas copiando soluções na internet sem saber o que você fez quando alguém lhe pede para explicar o resultado obtido.

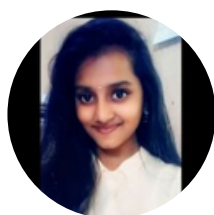
Este guia será o seu melhor amigo, e fará com que você queira praticar cada vez mais, pois você não caminhará sozinho, eu estarei lá para guiá-lo em caso de dúvidas.

## Objectivo do Ebook

O principal objetivo deste guia é fazer com que você pratique Python de forma mais prática, pois acredita,os que criar projetos é a melhor forma de praticar qualquer linguagem de programação.

## QUEM SOMOS NÓS ?

Aprendemos juntos, acreditamos no compartilhamento de conhecimento, capacitamos e inspiramos nossos alunos e nós mesmos a nos desenvolvermos por meio do poder da aprendizagem ao longo da vida. Trabalhamos duro para ter sucesso porque estamos muito comprometidos com nosso propósito de compartilhar conhecimento com qualquer pessoa ao redor do mundo.



Rajyalaxmi Singam

Criadora de conteúdo,  
Desenvolvedora Python,  
Instrutor de Programação



Joao Futi Muanda


Criador de conteúdo,  
Programador Python,  
Desenvolvedor Web,  
Instrutor de Programação

# Usando Python Vol.04 - Criando Projetos em Python

Neste guia de ebook, você não caminhará sozinho, pois estarei sempre disponível para orientá-lo e responder suas dúvidas (LIVE ou apenas por mensagens). Pois com este guia, quero que você pratique a linguagem de programação Python de uma forma mais prática que é criar projetos em Python, e também haverá projetos que você pode colocar no seu portfólio ou no seu GitHub.

Nesta edição você criará o seguinte projeto:

## Inventário Doméstico



### Inventário Doméstico

Nome

Sala/Área

Descrição

Marca/Modelo

Data da compra

Valor da compra

Número de série

Imagem do item

CARREGAR

+ ADICIONAR

↺ ATUALIZAR

🗑 DELETAR

CONFIRMAR


📄 VER ITEM

Valor Total de todos os itens

R\$ 15,436.00

Quantidade total de itens

3



#Item	Nome	Sala/Área	Descrição	Marca/Modelo	Data Da Compra	Valor Da Compra	Número De Série
1	Baldezinho	Cozinha	comprei em 2020	ef55	4/16/20	5006	226q5q5
2	Sofa	Sala de estar	Sofa antigo	Sofa	4/13/18	1506	s5553
4	teste	Cuzinha	comprei a um bom tempo	Mdk	6/24/20	8924	defe

O guia está bem estruturado e comentado, e em caso de dúvida pode sempre contactar-nos para esclarecimentos.

Este projeto será muito útil e também será útil para o seu **portfólio** sem dúvida. Aqui eu gostaria que você não apenas seguisse o tutorial, mas também ao longo do caminho para adicionar novos recursos, ou melhor, deixar sua criatividade fluir.

**Importante** : Após a compra deste produto, poderei ajudá-lo por 1 mês a entender o conteúdo aqui, então fique à vontade e não hesite em entrar em contato comigo se tiver alguma dúvida ao longo do processo.

**Nota:** neste guia em alguns pontos você precisará entrar em contato comigo para que eu possa explicar alguns dos códigos aqui presentes, pois apenas lendo não será fácil de entender devido à complexidade de certos códigos aqui presentes, então em caso de dúvida recomendo que entre em contato comigo pelo instagram ou linkedin, pois lá posso te responder sem demora.

### **O que é Inventário Doméstico ?**

Inventário Doméstico é uma classe de aplicativos de software de rastreamento de ativos que permite catalogar os detalhes importantes de todos os seus bens pessoais. Ele rastreia os locais e o valor de cada um de seus itens com o clique de um mouse. Isto é o que vamos construir aqui neste tutorial

## Criando Banco de dados

Como esta aplicação terá uma conexão com um banco de dados, então primeiro vamos começar a criar nosso banco de dados, este será um banco de dados bem simples sem muitas tabelas, apenas uma, então vamos começar a criar nosso banco de dados.

Crie um novo arquivo Python e salve-o como **criarbd.py** e importe o Sqlite dentro desse arquivo.

```
# importando o SQLite
import sqlite3 as lite
```

Agora vamos criar uma conexão com um novo banco de dados que se parecerá com isto:

```
# Criando conexão
con = lite.connect('dados.db')
```

Após criar a conexão, criaremos nosso banco de dados.

```
# Criando tabela
with con:
    cur = con.cursor()
    cur.execute("CREATE TABLE Inventario(id INTEGER PRIMARY KEY AUTOINCREMENT,nome TEXT,
local TEXT, descricao TEXT,marca TEXT, data_da_compra DATE, valor_da_compra DECIMAL,
serie TEXT, imagem TEXT)")
```

Agora execute o código que nosso banco de dados será criado, desta forma acabamos finalizando a primeira fase do projeto

## Criando operações para o banco de dados (CRUD)

Depois de criar nosso banco de dados, vamos criar funções que nos permitirão interagir com o banco de dados criando um novo arquivo python que chamarei de **view.py**, essas funções serão:

```

import sqlite3 as lite
from datetime import datetime

# Criando conexão
con = lite.connect('dados.db')

# Inserir inventario
def inserir_form(i):
    with con:
        cur = con.cursor()
        query = "INSERT INTO Inventario (nome, local, descricao, marca, data_da_compra,
        valor_da_compra, serie, imagem) VALUES (?, ?, ?, ?, ?, ?, ?, ?)"
        cur.execute(query, i)

# Deletar inventario
def deletar_form(i):
    with con:
        cur = con.cursor()
        query = "DELETE FROM Inventario WHERE id=?"
        cur.execute(query, i)

# Atualizar inventario
def atualizar_form(i):
    with con:
        cur = con.cursor()
        query = "UPDATE Inventario SET nome=?, local=?, descricao=?, marca=?,
        data_da_compra=?, valor_da_compra=?, serie=?, imagem=? WHERE id=?"
        cur.execute(query, i)

# Ver Inventario
def ver_form():
    lista_itens = []
    with con:
        cur = con.cursor()
        cur.execute("SELECT * FROM Inventario")
        rows = cur.fetchall()
        for row in rows:
            lista_itens.append(row)
        return lista_itens

# Ver Iten no inventario
def ver_iten(id):
    lista_itens = []
    with con:
        cur = con.cursor()
        cur.execute("SELECT * FROM Inventario WHERE id=?", (id))
        rows = cur.fetchall()
        for row in rows:
            lista_itens.append(row)
        return lista_itens

```



Como eu disse neste guia, você não fará isso sozinho quando chegar a esta parte, sintá-se à vontade para entrar em contato comigo e explicarei essa parte para você.

## Criando a interface

Ok, agora que a parte de backend está pronta, vamos criar a interface para nossa aplicação, começar criando um novo arquivo, o meu vou nomeá-lo main.py, e começar importando as seguintes bibliotecas para o novo script.

```
from tkinter import*
from tkinter import Tk, StringVar, ttk
import tkinter.font as tkFont
from tkinter import messagebox
from tkinter import filedialog as fd
```

Vamos agora criar uma janela vazia, e também existem algumas cores que selecionei que usaremos neste projeto, então você passará o seguinte código para o seu arquivo:

```
##### cores #####

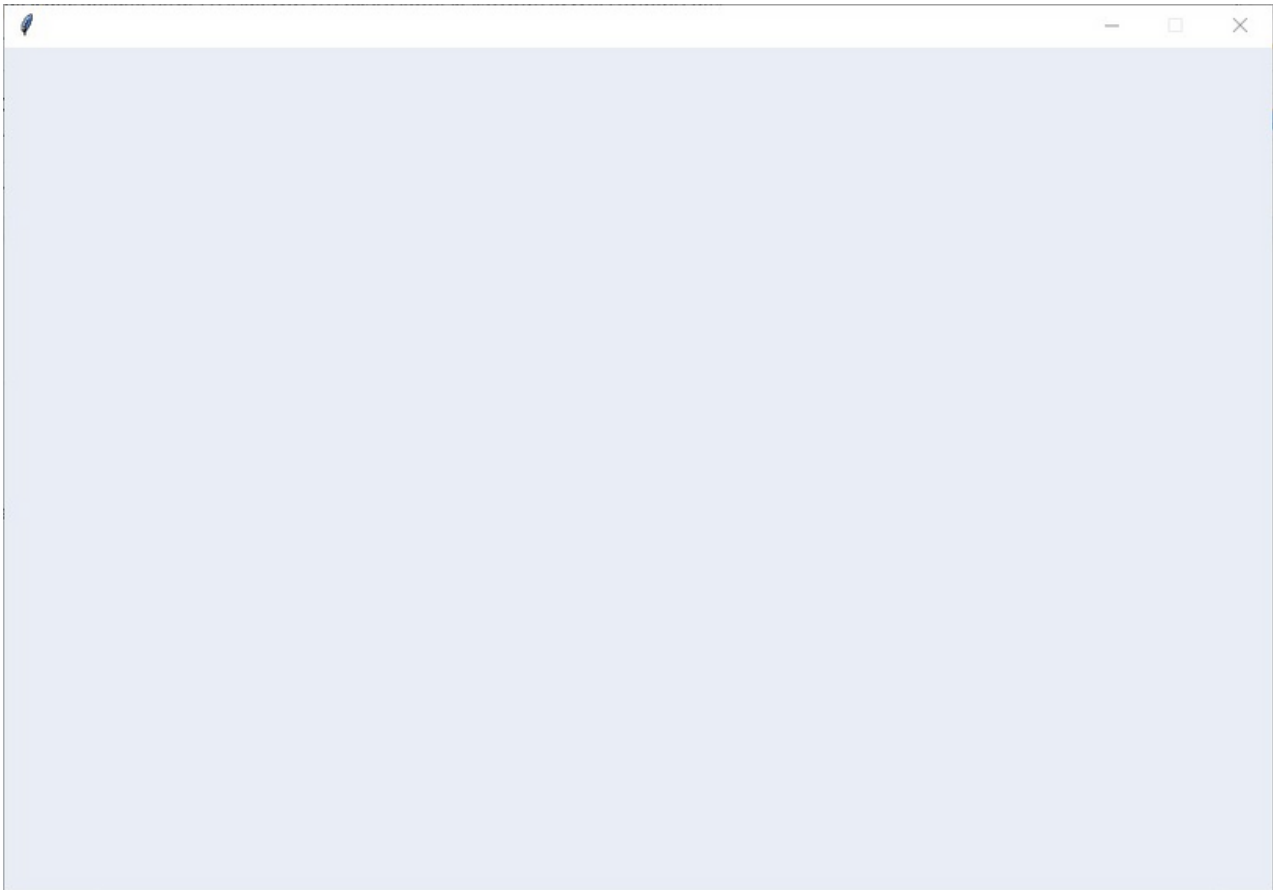
co0 = "#2e2d2b" # Preta
co1 = "#feffff" # branca
co2 = "#4fa882" # verde
co3 = "#38576b" # valor
co4 = "#403d3d" # letra
co5 = "#e06636" # - profit
co6 = "#038cfc" # azul
co7 = "#3fbfb9" # verde
co8 = "#263238" # + verde
co9 = "#e9edf5" # + verde

##### criando janela #####

janela = Tk ()
janela.title ("")
janela.geometry('900x600')
janela.configure(background=co9)
janela.resizable(width=FALSE, height=FALSE)

style = ttk.Style(janela)
style.theme_use("clam")

janela.mainloop()
```



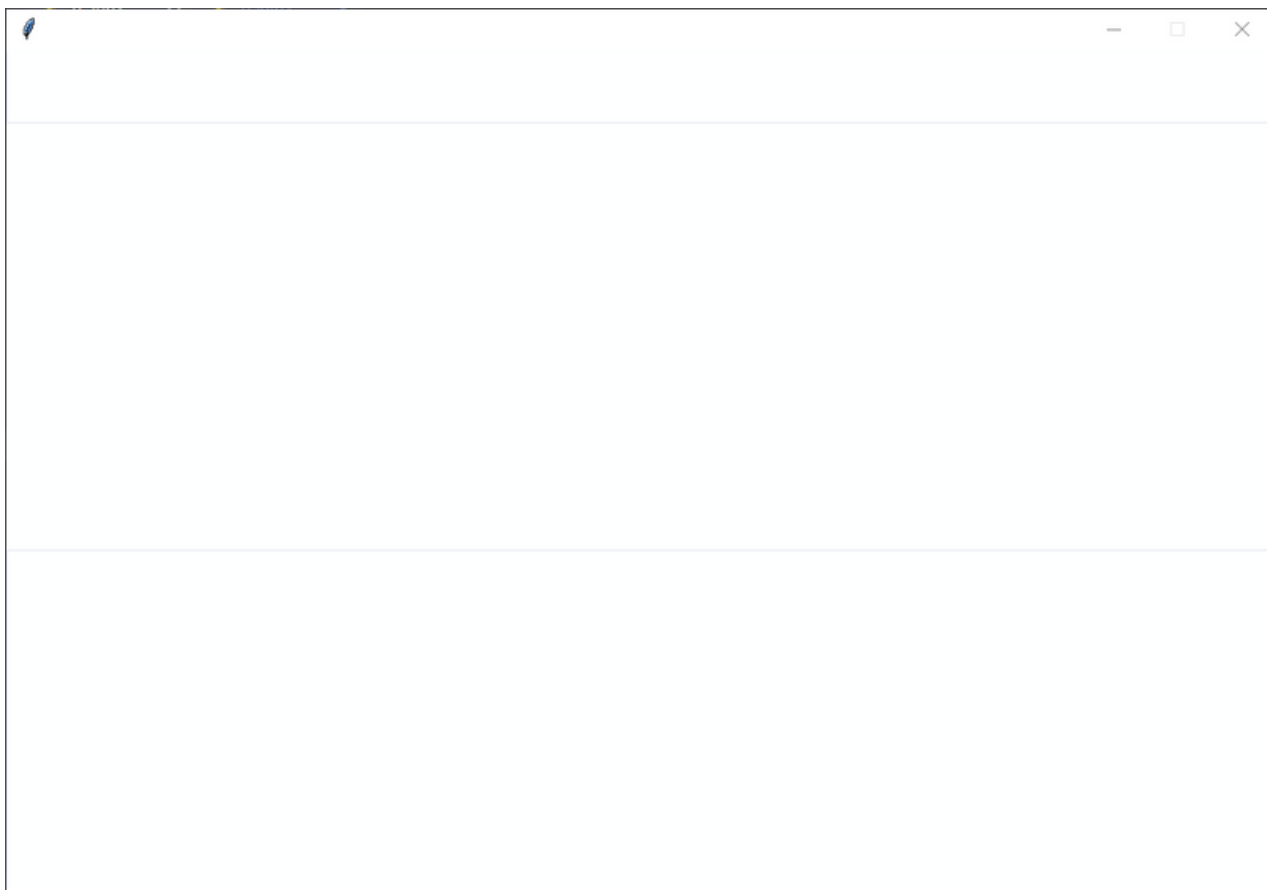
Agora vamos criar alguns FRAMES que nos permitirão dividir a janela que acabamos de criar:

```
##### Frames #####

frameCima = Frame(janela, width=1043, height=50, bg=co1, relief="flat",)
frameCima.grid(row=0, column=0)

frameMeio = Frame(janela,width=1043, height=303,bg=co1, pady=20, relief="flat")
frameMeio.grid(row=1, column=0,pady=1, padx=0, sticky=NSEW)

frameDireita = Frame(janela,width=1043, height =300,bg=co1, relief="flat")
frameDireita.grid(row=2, column=0,pady=0,padx=1, sticky=NSEW)
```



Agora vamos começar colocando o logo do app, que será algo assim:

```
# abrindo imagem
app_img = Image.open('inventorio.png')
app_img = app_img.resize((45, 45))
app_img = ImageTk.PhotoImage(app_img)

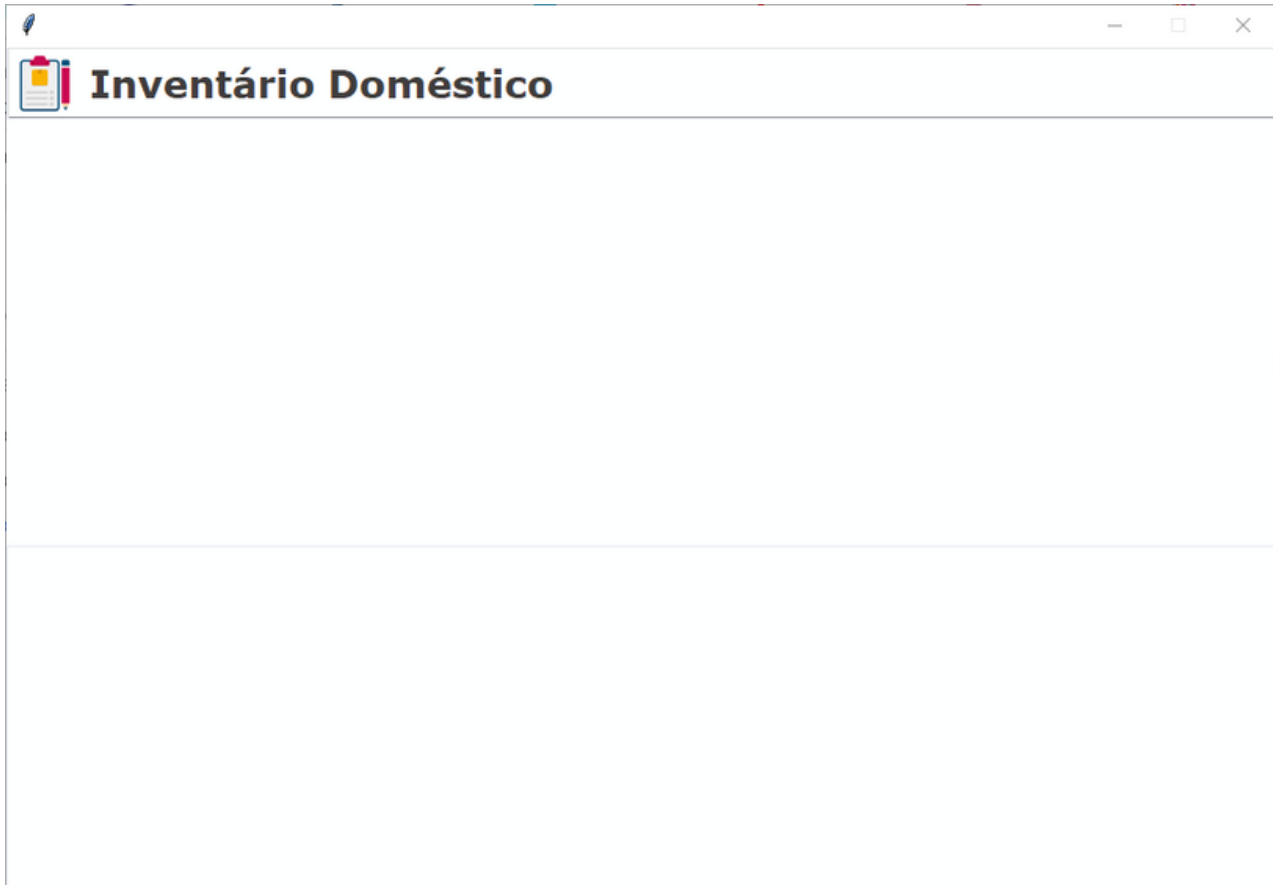
app_logo = Label(frameCima,image=app_img,text=" Inventário Doméstico",    width=900,
compound=LEFT, relief=RAISED,anchor=NW, font=('Verdana 20 bold'),bg=co1,fg =co4 )
app_logo.place(x=0, y=0)
```

Veja, se você executar este aplicativo você receberá um erro porque o Pillow não está instalado, então você terá que instalar a biblioteca Pillow, basta fazer o seguinte no seu terminal, digite: **pip install pillow**

E depois importa o pillow para o seu script

```
from PIL import Image, ImageTk
```

Veja para imagens use o site <https://icons8.com/icons/set/angola> pois existem várias imagens que você pode usar em seu projeto.



Perfeito, depois disso vamos criar alguns campos de entrada que serão os seguintes:

```

# criando entradas
l_nome = Label(frameMeio, text="Nome", height=1, anchor=NW, font=('Ivy 10 bold'), bg=co1, fg=co4)
l_nome.place(x=10, y=10)
e_nome = Entry(frameMeio, width=30, justify='left', relief="solid")
e_nome.place(x=130, y=11)

l_local = Label(frameMeio, text="Sala/Área", height=1, anchor=NW, font=('Ivy 10 bold'), bg=co1, fg=co4)
l_local.place(x=10, y=40)
e_local = Entry(frameMeio, width=30, justify='left', relief="solid")
e_local.place(x=130, y=41)

l_descricao = Label(frameMeio, text="Descrição", height=1, anchor=NW, font=('Ivy 10 bold'), bg=co1, fg=co4)
l_descricao.place(x=10, y=70)
e_descricao = Entry(frameMeio, width=30, justify='left', relief="solid")
e_descricao.place(x=130, y=71)

l_model = Label(frameMeio, text="Marca/Modelo", height=1, anchor=NW, font=('Ivy 10 bold'), bg=co1, fg=co4)
l_model.place(x=10, y=100)
e_model = Entry(frameMeio, width=30, justify='left', relief="solid")
e_model.place(x=130, y=101)

l_cal = Label(frameMeio, text="Data da compra", height=1, anchor=NW, font=('Ivy 10 bold'), bg=co1, fg=co4)
l_cal.place(x=10, y=130)
e_cal = DateEntry(frameMeio, width=12, background='darkblue', foreground='white', borderwidth=2, year=2020)
e_cal.place(x=130, y=131)

l_valor = Label(frameMeio, text="Valor da compra", height=1, anchor=NW, font=('Ivy 10 bold'), bg=co1,
fg=co4)
l_valor.place(x=10, y=160)
e_valor = Entry(frameMeio, width=30, justify='left', relief="solid")
e_valor.place(x=130, y=161)

l_serial = Label(frameMeio, text="Número de série", height=1, anchor=NW, font=('Ivy 10 bold'), bg=co1,
fg=co4)
l_serial.place(x=10, y=190)
e_serial = Entry(frameMeio, width=30, justify='left', relief='solid')
e_serial.place(x=130, y=191)

```

Veja, assim que você executar este código, você receberá um erro, isso é devido a uma biblioteca ausente. Então, para corrigir isso, basta instalar a seguinte biblioteca e importá-la para o seu Script.

```

pip install tkcalendar

```

— □ ×



## Inventário Doméstico

Nome

Sala/Área

Descrição

Marca/Modelo

Data da compra

Valor da compra

Número de série

Agora vamos criar os botões que irão realizar as funções CRUD para que tenhamos o seguinte.

```

l_carregar = Label(frameMeio, text="Imagem do item", height=1, anchor=NW, font=('Ivy 10 bold'),
bg=co1, fg=co4)
l_carregar.place(x=10, y=220)
botao_carregar = Button(frameMeio, compound=CENTER, anchor=CENTER, text="carregar".upper(),
width=30, overrelief=RIDGE, font=('ivy 8'), bg=co1, fg=co0 )
botao_carregar.place(x=130, y=221)

# Botao Inserir
img_add = Image.open('add.png')
img_add = img_add.resize((20, 20))
img_add = ImageTk.PhotoImage(img_add)
botao_inserir = Button(frameMeio, image=img_add, compound=LEFT, anchor=NW, text="
Adicionar".upper(), width=95, overrelief=RIDGE, font=('ivy 8'), bg=co1, fg=co0 )
botao_inserir.place(x=330, y=10)

# Botao Atualizar
img_update = Image.open('update.png')
img_update = img_update.resize((20, 20))
img_update = ImageTk.PhotoImage(img_update)
botao_atualizar = Button(frameMeio, image=img_update, compound=LEFT, anchor=NW, text="
Atualizar".upper(), width=95, overrelief=RIDGE, font=('ivy 8'), bg=co1, fg=co0 )
botao_atualizar.place(x=330, y=50)

# Botao Deletar
img_delete = Image.open('delete.png')
img_delete = img_delete.resize((20, 20))
img_delete = ImageTk.PhotoImage(img_delete)
botao_deletar = Button(frameMeio, image=img_delete, compound=LEFT, anchor=NW, text="
Deletar".upper(), width=95, overrelief=RIDGE, font=('ivy 8'), bg=co1, fg=co0 )
botao_deletar.place(x=330, y=90)

# Botao ver Item
img_item = Image.open('item.png')
img_item = img_item.resize((20, 20))
img_item = ImageTk.PhotoImage(img_item)
botao_ver = Button(frameMeio, image=img_item, compound=LEFT, anchor=NW, text="
Ver
item".upper(), width=95, overrelief=RIDGE, font=('ivy 8'), bg=co1, fg=co0 )
botao_ver.place(x=330, y=221)

# Labels Quantidade total e Valores
l_total = Label(frameMeio, width=14, height=2, anchor=CENTER, font=('Ivy 17 bold'), bg=co7, fg=co1,
relief=FLAT)
l_total.place(x=450, y=17)
l_valor_total = Label(frameMeio, text=' Valor Total de todos os itens ', anchor=NW, font=('Ivy 10 bold'),
bg=co7, fg=co1)
l_valor_total.place(x=450, y=12)

l_qtd = Label(frameMeio, width=10, height=2, anchor=CENTER, font=('Ivy 25 bold'), bg=co7, fg=co1,
relief=FLAT)
l_qtd.place(x=450, y=90)
l_qtd_itens = Label(frameMeio, text='Quantidade total de itens', anchor=NW, font=('Ivy 10 bold'), bg=co7,
fg=co1)
l_qtd_itens.place(x=460, y=92)

```





## Inventário Doméstico

Nome

Sala/Área

Descrição

Marca/Modelo

Data da compra

7/23/20

Valor da compra

Número de série

Imagem do item

CARREGAR

+

ADICIONAR

↺

ATUALIZAR

🗑

DELETAR

📄

VER ITEM

Valor Total de todos os itens

Quantidade total de itens

## Criando Tabela

Veja, assim que chegar nessa parte, aconselho você a entrar em contato comigo para que eu possa te explicar, mas se você entender o que foi feito, então pode seguir em frente.

Muito bem, depois disso vamos agora criar uma tabela que apresentará os dados presentes no banco de dados.

Para isso usaremos o widget treeview, será algo como o abaixo (copie este mesmo código, ou você também pode escrevê-los linha por linha para entender o que está sendo feito)



```

# funcao para mostrar
def mostrar():

    # creating a treeview with dual scrollbars
    tabela_head = ['#Item', 'Nome', 'Sala/Área', 'Descrição', 'Marca/Modelo', 'Data da compra', 'Valor da compra', 'Número de série']

    lista_itens = []

    global tree

    tree = ttk.Treeview(frameDireita, selectmode="extended",
        columns=tabela_head, show="headings")
    # vertical scrollbar
    vsb = ttk.Scrollbar(
        frameDireita, orient="vertical", command=tree.yview)
    # horizontal scrollbar
    hsb = ttk.Scrollbar(
        frameDireita, orient="horizontal", command=tree.xview)

    tree.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)

    tree.grid(column=0, row=0, sticky='nsew')
    vsb.grid(column=1, row=0, sticky='ns')
    hsb.grid(column=0, row=1, sticky='ew')
    frameDireita.grid_rowconfigure(0, weight=12)

    hd=["center", "center", "center", "center", "center", "center", "center", 'center']
    h=[40,150,100,160,130,100,100, 100]

    n=0

    for col in tabela_head:
        tree.heading(col, text=col.title(), anchor=CENTER)
        # adjust the column's width to the header string
        tree.column(col, width=h[n], anchor=hd[n])

    n+=1

    for item in lista_itens:
        tree.insert('', 'end', values=item)

    quantidade = []
    for iten in lista_itens:
        quantidade.append(iten[6])

    Total_valor = sum(quantidade)
    Total_itens = len(quantidade)

    l_total['text'] = 'R$ {:.2f}'.format(Total_valor)
    l_qtd['text'] = Total_itens

    mostrar()

```

Assim terminamos a parte do desenho, agora vamos dar vida ao nosso projeto.

## Criando funções

Vamos começar criando primeiro a função para adicionar itens ao banco de dados.

Veja que devido a complexidade de determinados códigos recomendo que entre em contato comigo para que eu possa te explicar essas funções, pois aqui você terá apenas o código das funções e não a explicação completa, pois seria difícil entender apenas lendo o código.

## Função Inserir

No código atual, vá abaixo onde você tem o seguinte código:

```
app_logo = Label(frameCima, image=app_img, text=" Inventário Doméstico", width=900,
compound=LEFT, relief=RAISED, anchor=NW, font=('Verdana 20 bold'),bg=co1, fg=co4 )
app_logo.place(x=0, y=0)
```

Vamos criar todas as funções neste código.

Agora veja, o que vamos fazer primeiro é criar uma variável global chamada tree, que é o nome da tabela que criamos dentro da função show, para que possamos ter essa tabela em qualquer parte do nosso código, principalmente as funções que vamos criar.

```
global tree
```

Agora vamos criar nossa função insert que será como o código abaixo:

```
# funcao inserir
def inserir():
    global imagem,imagem_string, l_imagem
    nome = e_nome.get()
    local = e_local.get()
    descricao = e_descricao.get()
    model = e_model.get()
    data = e_cal.get()
    valor = e_valor.get()
    serie = e_serial.get()
    imagem = imagem_string

    lista_inserir = [nome, local, descricao, model, data, valor, serie,imagem]

    for i in lista_inserir:
        if i=='':
            messagebox.showerror('Erro', 'Preencha todos os campos')
            return

    inserir_form(lista_inserir)

    messagebox.showinfo(
        'Sucesso', 'Os dados foram inseridos com sucesso')

    e_nome.delete(0, 'end')
    e_local.delete(0, 'end')
    e_descricao.delete(0, 'end')
    e_model.delete(0, 'end')
    e_cal.delete(0, 'end')
    e_valor.delete(0, 'end')
    e_serial.delete(0, 'end')

    for widget in frameDireita.winfo_children():
        widget.destroy()

    mostrar()
```

## Função atualizar

Abaixo da função inserir você colocará o seguinte código:

```
# funcao atualizar
def atualizar():
    try:
        treev_dados = tree.focus()
        treev_dicionario = tree.item(treev_dados)  treev_lista =
treev_dicionario['values']
        valor = treev_lista[0]

e_nome.delete(0, 'end')
e_local.delete(0, 'end')
e_descricao.delete(0, 'end')
e_model.delete(0, 'end')
e_cal.delete(0, 'end')
e_valor.delete(0, 'end')
e_serial.delete(0, 'end')

id = int(treev_lista[0])
e_nome.insert(0, treev_lista[1])
e_local.insert(0, treev_lista[2])
e_descricao.insert(0, treev_lista[3])
e_model.insert(0, treev_lista[4])
e_cal.insert(0, treev_lista[5])
e_valor.insert(0, treev_lista[6])
e_serial.insert(0, treev_lista[7])

def update():
    global imagem,imagem_string, l_imagem
    nome = e_nome.get()
    local = e_local.get()
    descricao = e_descricao.get()
    model = e_model.get()
    data = e_cal.get()
    valor = e_valor.get()
    serie = e_serial.get()
    imagem = imagem_string
```

```

    if imagem == '':
        imagem = e_serial.insert(0, treev_lista[7])

lista_atualizar = [nome, local, descricao, model, data, valor,
serie,imagem, id]

for i in lista_atualizar:
    if i=='':
        messagebox.showerror('Erro', 'Preencha todos os campos') return

atualizar_form(lista_atualizar)

    messagebox.showinfo('Sucesso', 'Os dados foram atualizados com
sucesso')

e_nome.delete(0, 'end')
e_local.delete(0, 'end')
e_descricao.delete(0, 'end')
e_model.delete(0, 'end')
e_cal.delete(0, 'end')
e_valor.delete(0, 'end')
e_serial.delete(0, 'end')

botao_confirmar.destroy()

for widget in frameDireita.wininfo_children():
    widget.destroy()

    mostrar()
    botao_confirmar = Button(frameMeio, command=update,
text="Confirmar".upper(), width=13, height=1, bg=co2, fg=co1,font=('ivy 8
bold'),relief=RAISED, overrelief=RIDGE)
    botao_confirmar.place(x=330, y=185)

except IndexError:
    messagebox.showerror(
'Erro', 'Seleciona um dos dados na tabela')

```

## Função deletar

Abaixo da função atualizar você colocará o seguinte código:

```
# funcao deletar
def deletar():
    try:
        treev_dados = tree.focus()
        treev_dicionario = tree.item(treev_dados)
        treev_lista = treev_dicionario['values']
        valor = treev_lista[0]

        deletar_form([valor])

        messagebox.showinfo(
            'Sucesso', 'Os dados foram deletados com sucesso')

        for widget in frameDireita.winfo_children():
            widget.destroy()

        mostrar()

    except IndexError:
        messagebox.showerror(
            'Erro', 'Selecione um dos dados na tabela')
```

Muito bem, se você conseguir implementar com sucesso essas funções acima, a aplicação estará praticamente completa com apenas uma função faltando.

## Função abrir imagem

Abaixo da função deletar você colocará o seguinte código:

```
# funcao para abrir imagem
def ver_imagem():
    global l_imagem, imagem, imagem_string

    treev_dados = tree.focus()
    treev_dicionario = tree.item(treev_dados)
    treev_lista = treev_dicionario['values']
    valor = [int(treev_lista[0])]

    iten = ver_iten(valor)

    imagem = iten[0][8]

    # abrindo a imagem
    imagem = Image.open(imagem)
    imagem = imagem.resize((170, 170))
    imagem = ImageTk.PhotoImage(imagem)

    l_imagem = Label(frameMeio, image=imagem)
    l_imagem.place(x=700, y=10)
```

Parabéns, você conseguiu concluir o projeto, agora aconselho a implementar novas funções para este mesmo projeto, como um campo de pesquisa de produtos.

Tente usar sua criatividade para deixar o app muito mais atrativo e digno de ser colocado em seu portfólio.