

Criando uma pequena aplicação com Banco de dados

Agora vamos criar uma pequena aplicação com os comandos já vistos até agora e mais alguns que veremos durante o desenvolvimento de nossa aplicação com acesso ao banco de dados.

Para acessarmos o mesmo usaremos o módulo DBI (DataBase Interface), que usa uma API para se comunicar com o banco de dados.

No exemplo usaremos o SQLite, que já vem incorporado ao módulo DBI. Então podemos acessá-lo sem necessidade de instalar módulos externos.

O SQLite cria um arquivo com extensão .db em disco, contendo todas as tabelas da aplicação.

Vamos à estrutura da nossa aplicação, onde criaremos três arquivos:

App.py (Conterá os containeres da interface e executará a aplicação);

Usuarios.py (Classe de modelo para usuários);

Banco.py (Classe de banco de dados).

Começaremos pelo nosso arquivo Banco.py. Primeiro devemos importar o módulo do SQLite:

```
#importando módulo do SQLite
```

```
Import sqlite3
```

Após isso, seguimos com o restante do código do arquivo, conforme mostra a Listagem 8.

```
class Banco():
```

```
def __init__(self):
```

```
    self.conexao = sqlite3.connect('banco.db')
```

```
    self.createTable()
```

```
def createTable(self):
```

```
c = self.conexao.cursor()

c.execute("""create table if not exists usuarios (
    idusuario integer primary key autoincrement ,
    nome text,
    telefone text,
    email text,
    usuario text,
    senha text)""")

self.conexao.commit()

c.close()
```

Listagem 8. Banco.py

Já criamos nosso arquivo de conexão com o banco de dados e agora criaremos o arquivo usuarios.py, conforme mostra a Listagem 9.

```
from Banco import Banco
```

```
class Usuarios(object):
```

```
def __init__(self, idusuario = 0, nome = "", telefone = "",
email = "", usuario = "", senha = ""):

    self.info = {}

    self.idusuario = idusuario

    self.nome = nome

    self.telefone = telefone

    self.email = email

    self.usuario = usuario

    self.senha = senha
```

```
def insertUser(self):

    banco = Banco()

    try:

        c = banco.conexao.cursor()

        c.execute("insert into usuarios (nome, telefone, email,
        usuario, senha) values ('" + self.nome + "', '" +
        self.telefone + "', '" + self.email + "', '" +
        self.usuario + "', '" + self.senha + "')")

        banco.conexao.commit()
        c.close()

        return "Usuário cadastrado com sucesso!"
    except:
        return "Ocorreu um erro na inserção do usuário"

def updateUser(self):

    banco = Banco()

    try:

        c = banco.conexao.cursor()

        c.execute("update usuarios set nome = '" + self.nome + "',
        telefone = '" + self.telefone + "', email = '" + self.email +
        "', usuario = '" + self.usuario + "', senha = '" + self.senha +
        "' where idusuario = " + self.idusuario + " ")
```

```
banco.conexao.commit()
```

```
c.close()
```

```
return "Usuário atualizado com sucesso!"
```

```
except:
```

```
    return "Ocorreu um erro na alteração do usuário"
```

```
def deleteUser(self):
```

```
    banco = Banco()
```

```
    try:
```

```
        c = banco.conexao.cursor()
```

```
        c.execute("delete from usuarios where idusuario = " + self.idusuario + " ")
```

```
        banco.conexao.commit()
```

```
        c.close()
```

```
        return "Usuário excluído com sucesso!"
```

```
    except:
```

```
        return "Ocorreu um erro na exclusão do usuário"
```

```
def selectUser(self, idusuario):
```

```
    banco = Banco()
```

```
    try:
```

```
        c = banco.conexao.cursor()
```

```
        c.execute("select * from usuarios where idusuario = " + idusuario + " ")
```

```

for linha in c:

    self.idusuario = linha[0]

    self.nome = linha[1]

    self.telefone = linha[2]

    self.email = linha[3]

    self.usuario = linha[4]

    self.senha = linha[5]


c.close()


return "Busca feita com sucesso!"

except:

    return "Ocorreu um erro na busca do usuário"

```

Listagem 9. Usuarios.py

Já temos a classe de conexão com o banco de dados e a classe que modela o usuário.

O próximo passo é utilizar os dados dessas duas classes no nosso arquivo de execução App.py, que terá a nossa interface, como mostra a Listagem 10.

```

from Usuarios import Usuarios

from tkinter import *


class Application:

    def __init__(self, master=None):

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


        self.container1 = Frame(master)

        self.container1["pady"] = 10

        self.container1.pack()

        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()
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.container8 = Frame(master)
self.container8["padx"] = 20
self.container8["pady"] = 10
self.container8.pack()
self.container9 = Frame(master)
self.container9["pady"] = 15
self.container9.pack()

self.titulo = Label(self.container1, text="Informe os dados :")
```

```
self.titulo["font"] = ("Calibri", "9", "bold")  
self.titulo.pack ()
```

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

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

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

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

```
self.txttelefone = Entry(self.container4)
```

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

```
self.lblemail= Label(self.container5, text="E-mail:",  
font=self.fonte, width=10)  
self.lblemail.pack(side=LEFT)
```

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

```
self.lblusuario= Label(self.container6, text="Usuário:",  
font=self.fonte, width=10)  
self.lblusuario.pack(side=LEFT)
```

```
self.txtusuario = Entry(self.container6)  
self.txtusuario["width"] = 25  
self.txtusuario["font"] = self.fonte  
self.txtusuario.pack(side=LEFT)
```

```
self.lblsenha= Label(self.container7, text="Senha:",  
font=self.fonte, width=10)  
self.lblsenha.pack(side=LEFT)
```

```
self.txtsenha = Entry(self.container7)  
self.txtsenha["width"] = 25  
self.txtsenha["show"] = "*"   
self.txtsenha["font"] = self.fonte  
self.txtsenha.pack(side=LEFT)
```



```
self.bntInsert = Button(self.container8, text="Inserir",  
font=self.fonte, width=12  
self.bntInsert["command"] = self.inserirUsuario  
self.bntInsert.pack (side=LEFT)
```

```
self.bntAlterar = Button(self.container8, text="Alterar",  
font=self.fonte, width=12)  
self.bntAlterar["command"] = self.alterarUsuario  
self.bntAlterar.pack (side=LEFT)
```

```
self.bntExcluir = Button(self.container8, text="Excluir",  
font=self.fonte, width=12)  
self.bntExcluir["command"] = self.excluirUsuario  
self.bntExcluir.pack(side=LEFT)
```

```
self.lblmsg = Label(self.container9, text="")  
self.lblmsg["font"] = ("Verdana", "9", "italic")  
self.lblmsg.pack()
```

```
def inserirUsuario(self):
```

```
    user = Usuarios()
```

```
    user.nome = self.txtnome.get()
```

```
    user.telefone = self.txttelefone.get()
```

```
    user.email = self.txtemail.get()
```

```
    user.usuario = self.txtusuario.get()
```

```
    user.senha = self.txtsenha.get()
```

```
    self.lblmsg["text"] = user.insertUser()
```

```
self.txtidusuario.delete(0, END)
self.txtnome.delete(0, END)
self.txttelefone.delete(0, END)
self.txtemail.delete(0, END)
self.txtusuario.delete(0, END)
self.txtsenha.delete(0, END)
```

```
def alterarUsuario(self):
```

```
    user = Usuarios()
```

```
    user.idusuario = self.txtidusuario.get()
```

```
    user.nome = self.txtnome.get()
```

```
    user.telefone = self.txttelefone.get()
```

```
    user.email = self.txtemail.get()
```

```
    user.usuario = self.txtusuario.get()
```

```
    user.senha = self.txtsenha.get()
```

```
    self.lblmsg["text"] = user.updateUser()
```

```
self.txtidusuario.delete(0, END)
```

```
self.txtnome.delete(0, END)
```

```
self.txttelefone.delete(0, END)
```

```
self.txtemail.delete(0, END)
```

```
self.txtusuario.delete(0, END)
```

```
self.txtsenha.delete(0, END)
```

```
def excluirUsuario(self):  
    user = Usuarios()  
  
    user.idusuario = self.txtidusuario.get()  
  
    self.lblmsg["text"] = user.deleteUser()  
  
    self.txtidusuario.delete(0, END)  
    self.txtnome.delete(0, END)  
    self.txttelefone.delete(0, END)  
    self.txtemail.delete(0, END)  
    self.txtusuario.delete(0, END)  
    self.txtsenha.delete(0, END)  
  
def buscarUsuario(self):  
    user = Usuarios()  
  
    idusuario = self.txtidusuario.get()  
  
    self.lblmsg["text"] = user.selectUser(idusuario)  
  
    self.txtidusuario.delete(0, END)  
    self.txtidusuario.insert(INSERT, user.idusuario)  
  
    self.txtnome.delete(0, END)  
    self.txtnome.insert(INSERT, user.nome)  
  
    self.txttelefone.delete(0, END)  
    self.txttelefone.insert(INSERT, user.telefone)
```

```
self.txtemail.delete(0, END)
```

```
self.txtemail.insert(INSERT, user.email)
```

```
self.txtusuario.delete(0, END)
```

```
self.txtusuario.insert(INSERT, user.usuario)
```

```
self.txtsenha.delete(0, END)
```

```
self.txtsenha.insert(INSERT, user.senha)
```

```
root = Tk()
```

```
Application(root)
```

```
root.mainloop()
```

Listagem 10. App.py (Interface)

O resultado será a interface da Figura 9, que mostra basicamente um CRUD (Create – Read – Update - Delete).



Figura 9. Interface CRUD com Tkinter

No artigo foi mostrado uma **introdução ao Tkinter**, que foi escolhido justamente pela quantidade de recursos que oferece, por ser relativamente simples e por ser nativo no **pacote de instalação do Python**.