



TÉCNICAS DE PROGRAMAÇÃO II

PROFº MAYLON HENRIQUE DE OLIVEIRA

MONGO DB – BANCO DE DADOS NÃO RELACIONAL - NOSQL

Utilizando Mongo DB



MONGO DB – INSTALAÇÃO MONGO

1º Passo Instalar a biblioteca

pymongo digitar o comando no cmd do Windows.

Comando -> `pip install pymongo`

2º Digitação do Código

```
from tkinter import *  
from tkinter import ttk  
import tkinter as tk  
import pymongo
```

Biblioteca para
utilização do
combobox



Biblioteca do mongodb



MONGO DB – CONFIGURAÇÕES

Configuração da tela

```
tela = Tk()
tela.title("Exemplo Mongo DB")
tela.geometry("800x600")
tela.resizable(True, True)
tela.configure(background="#ffffff")
```

Configuração do mongodb

```
exemplo = pymongo.MongoClient("mongodb://localhost:27017/")
db = exemplo["exemplo"]
collection = db["clientes"]
```

MONGO DB – BANCO DE DADOS NÃO RELACIONAL



MONGO DB – CONSTRUÇÃO FORMULÁRIO

Exemplo Mongo DB

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:



```
lbl_codigo = Label(tela, text="Código:", bg="#ffffff").place(x=130, y=140)
txt_codigo = Entry(tela, width=20, borderwidth=2, fg="black", bg="white")
txt_codigo.place(x=190, y=140)

lbl_nome = Label(tela, text="Nome:", bg="#ffffff").place(x=130, y=170)
txt_nome = Entry(tela, width=40, borderwidth=2, fg="black", bg="white")
txt_nome.place(x=190, y=170)
txt_nome.insert(0, "")

lbl_cpf = Label(tela, text="CPF:", bg="#ffffff").place(x=450, y=170)
txt_cpf = Entry(tela, width=20, borderwidth=2, fg="black", bg="white")
txt_cpf.place(x=480, y=170)
txt_cpf.insert(0, "")
```

MONGO DB – CONSTRUÇÃO FORMULÁRIO

Exemplo Mongo DB

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:



```
lbl_idade = Label(tela, text="Idade:", bg="#ffffff").place(x=130, y=200)
txt_idade = Entry(tela, width=20, borderwidth=2, fg="black", bg="white")
txt_idade.place(x=190, y=200)
txt_idade.insert(0, "")
```

```
lbl_end = Label(tela, text="Rua:", bg="#ffffff").place(x=450, y=200)
txt_end = Entry(tela, width=25, borderwidth=2, fg="black", bg="white")
txt_end.place(x=480, y=200)
txt_end.insert(0, "")
```


MONGO DB – CONSTRUÇÃO FORMULÁRIO

Exemplo Mongo DB

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:

Bairro: Estado: Cidade:

```
lbl_bairro = Label(tela, text="Bairro:", bg="#ffffff").place(x= 130, y= 230)
txt_bairro = Entry(tela, width=20, borderwidth=2, fg="black", bg="white")
txt_bairro.place(x= 190, y=230)
txt_bairro.insert(0, " ")
```


MONGO DB – CONSTRUÇÃO FORMULÁRIO

Exemplo Mongo DB

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:

Bairro: Estado: Cidade:

```
lbl_estado = Label(tela, text="Estado:", bg="#ffffff").place(x=330, y=230)
comboestado = ttk.Combobox(tela,
                             values=[
                                 "São Paulo",
                                 "Rio de Janeiro",
                                 "Minas Gerais",
                                 "Espírito Santo"],)

comboestado.grid(column=0, row=1)
comboestado.place(x=370, y=230)
```

MONGO DB – CONSTRUÇÃO FORMULÁRIO

Exemplo Mongo DB

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:

Bairro: Estado: Cidade:

```
lbl_cidade = Label(tela, text="Cidade:", bg="#ffffff").place(x= 520, y= 230)
txt_cidade = Entry(tela, width=20, borderwidth=2, fg="black", bg="white")
txt_cidade.place(x= 570, y=230)
txt_cidade.insert(0, " ")
```

MONGO DB – CONSTRUÇÃO FORMULÁRIO

Exemplo Mongo DB

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:

Bairro: Estado: Cidade:

```
lbl_cidade = Label(tela, text="Cidade:", bg="#ffffff").place(x= 520, y= 230)
txt_cidade = Entry(tela, width=20, borderwidth=2, fg="black", bg="white")
txt_cidade.place(x= 570, y=230)
txt_cidade.insert(0, " ")
```

MONGO DB – FUNÇÃO SALVAR

```
def salvar():  
    codigo = txt_codigo.get()  
    nome = txt_nome.get()  
    idade = int(txt_idade.get())  
    end = txt_end.get()  
    cpf = txt_cpf.get()  
  
    bairro = txt_bairro.get()  
    cidade = txt_cidade.get()  
    estado = comboestado.get()
```



Cada variavel ligada a
uma caixa de texto

MONGO DB – LIMPANDO CAIXAS DE TEXTO DEPOIS DE SALVAR

```
txt_codigo.delete(0, tk.END)
txt_nome.delete(0, tk.END)
txt_idade.delete(0, tk.END)

txt_end.delete(0, tk.END)
txt_bairro.delete(0, tk.END)
txt_cidade.delete(0, tk.END)
comboestado.set("")
txt_cpf.delete(0, tk.END)
```



Após a inserção dos dados as caixas de textos, será apagada automaticamente

MONGO DB – INSERÇÃO DOS DADOS DICIONÁRIO DOS DADOS

```
cliente = {"código":codigo, "nome": nome, "idade":idade, "endereço": end, "cpf":cpf,  
           "bairro":bairro, "cidade":cidade, "estado": estado}  
collection.insert_one(cliente)
```



Dicionário chamado cliente está sendo criado. Este dicionário é usado para representar um conjunto de informações relacionadas a um cliente. Cada campo do cliente é representado como uma chave (por exemplo, "código", "nome", "idade", etc.) e o valor associado a cada chave é obtido de variáveis ou valores existentes no código, como codigo, nome, idade, etc.

collection é uma instância de um objeto que representa uma coleção de um banco de dados NoSQL (como o MongoDB). A função insert_one é chamada para inserir o dicionário cliente na coleção.

MONGO DB – ATUALIZAÇÃO DOS DADOS - UPDATE


```
def atualizar():  
    codigo = txt_codigo.get()  
    nome = txt_nome.get()  
    idade = int(txt_idade.get())  
  
    end = txt_end.get()  
    cpf = txt_cpf.get()  
  
    bairro = txt_bairro.get()  
    cidade = txt_cidade.get()  
    estado = comboestado.get()  
  
    collection.update_one({"código":codigo}, {"$set": {"código":codigo, "nome": nome,  
                                                       "idade":idade, "endereço": end, "cpf":cpf,  
                                                       "bairro":bairro,  
                                                       "cidade":cidade, "estado": estado}})
```



O método `update_one` é comumente usado em bancos de dados NoSQL, como o MongoDB, para atualizar um único documento que atende a um critério específico.

MONGO DB – ATUALIZAÇÃO DOS DADOS - UPDATE

```
def atualizar():  
    codigo = txt_codigo.get()  
    nome = txt_nome.get()  
    idade = int(txt_idade.get())  
  
    end = txt_end.get()  
    cpf = txt_cpf.get()  
  
    bairro = txt_bairro.get()  
    cidade = txt_cidade.get()  
    estado = comboestado.get()  
  
    collection.update_one({"código":codigo}, {"$set": {"código":codigo, "nome": nome,  
                                                       "idade":idade, "endereço": end, "cpf":cpf,  
                                                       "bairro":bairro,  
                                                       "cidade":cidade, "estado": estado}})
```



O segundo argumento é outro dicionário que usa a operação \$set para especificar quais campos e valores devem ser atualizados no registro correspondente, todos os campos estão sendo atualizados com os valores das variáveis correspondentes, como "código", "nome", "idade", "endereço", "cpf", "bairro", "cidade" e "estado".

MONGO DB – APAGAR OS DADOS - DELETE

```
def apagar():  
    codigo = txt_codigo.get()  
    collection.delete_one({"código": codigo})
```



Função utiliza a função delete_one para excluir um registro no banco de dados.

A exclusão é feita com base na correspondência do campo "código" do registro com o valor armazenado na variável codigo.

O único argumento para a função delete_one é um dicionário que especifica o critério de pesquisa. Neste caso, o critério é que o campo "código" no banco de dados corresponda ao valor da variável codigo.

MONGO DB – CRIANDO OS BOTÕES

Agora que já criamos as funções iremos criar os botões

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:

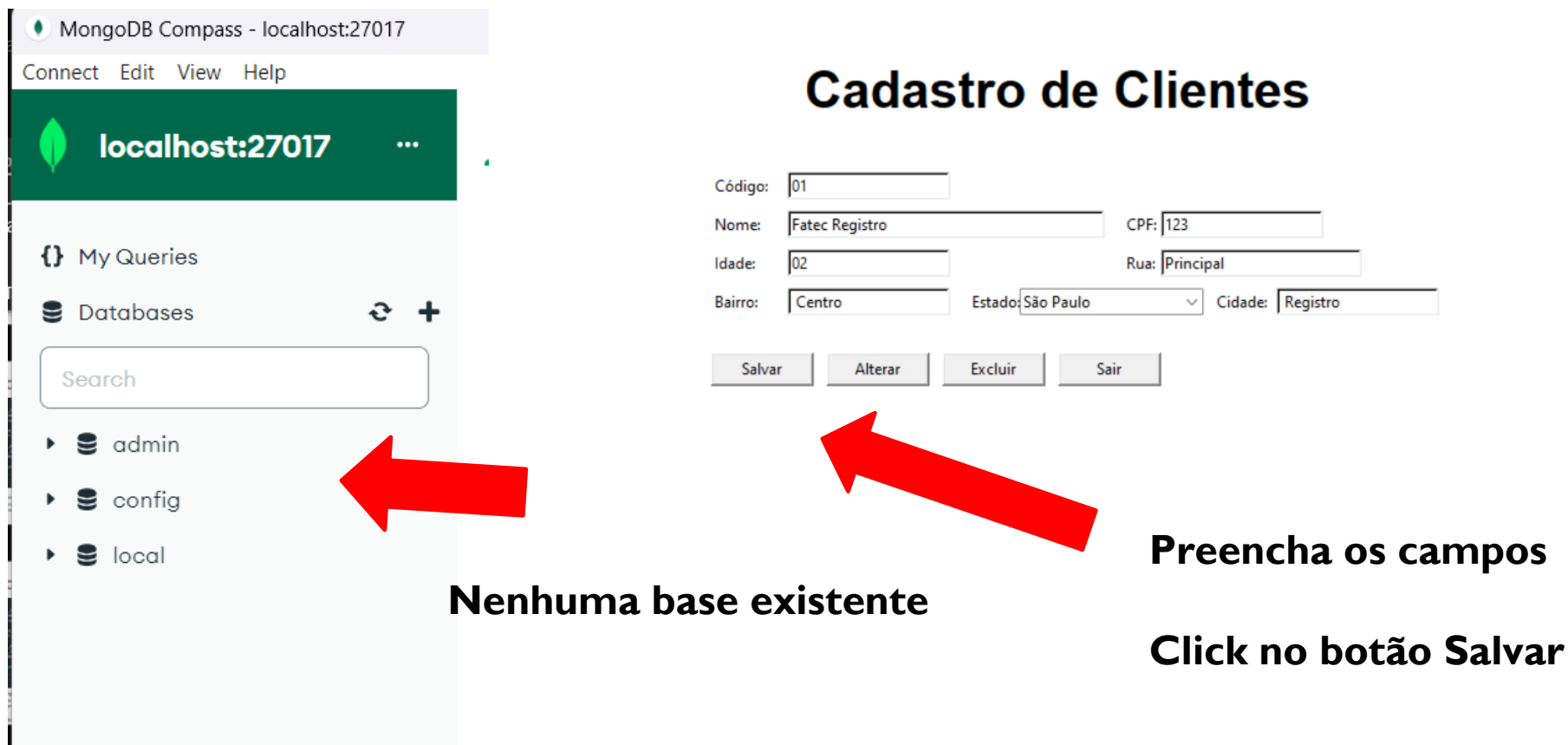
Bairro: Estado: Cidade:

```
btn_salvar = Button(tela, text="Salvar", width=10, command=salvar).place(x=130, y=280)
btn_alterar = Button(tela, text="Alterar", width=10, command=atualizar).place(x=220, y=280)
btn_excluir = Button(tela, text="Excluir", width=10, command=apagar).place(x=310, y=280)
btn_sair = Button(tela, text="Sair", width=10, command=tela.quit).place(x=400, y=280)
tela.mainloop()
```

Cada botão chama uma função

MONGO DB – REALIZANDO O TESTE DA APLICAÇÃO - SALVAR

Realizando o teste da aplicação - Salvar



The screenshot displays the MongoDB Compass application interface. On the left, a sidebar shows the 'Databases' section with a search bar and a list of databases: 'admin', 'config', and 'local'. A red arrow points from the text 'Nenhuma base existente' to this list. The main area on the right is titled 'Cadastro de Clientes' and contains a form with the following fields: 'Código' (01), 'Nome' (Fatec Registro), 'CPF' (123), 'Idade' (02), 'Rua' (Principal), 'Bairro' (Centro), 'Estado' (São Paulo), and 'Cidade' (Registro). Below the form are four buttons: 'Salvar', 'Alterar', 'Excluir', and 'Sair'. A red arrow points from the text 'Preencha os campos' to the form fields, and another red arrow points from the text 'Click no botão Salvar' to the 'Salvar' button.

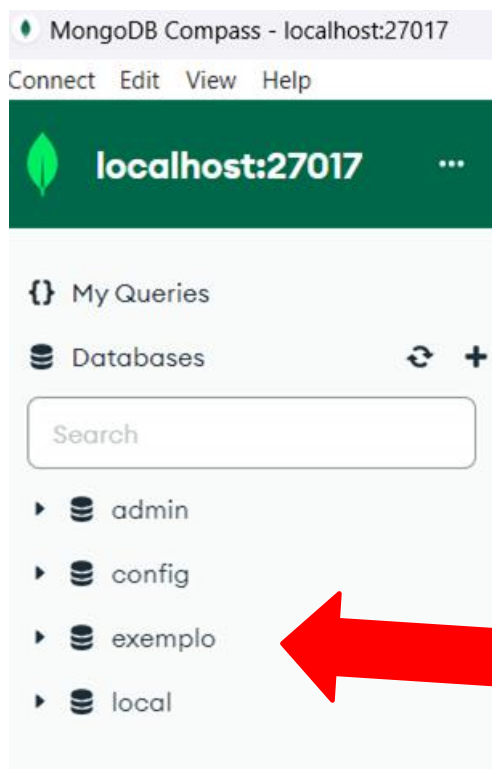
Nenhuma base existente

Preencha os campos

Click no botão Salvar

MONGO DB – REALIZANDO O TESTE DA APLICAÇÃO - SALVAR

Atualize o mongo db



Base

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:

Bairro: Estado: Cidade:

+ ADD DATA

EXPORT DATA

```
{
  "_id": ObjectId('654ac5f5e05d8af1616e400b'),
  "código": "01",
  "nome": "Fatec Registro",
  "idade": 2,
  "endereço": "Principal",
  "cpf": "123",
  "bairro": "Centro",
  "cidade": "Registro",
  "estado": "São Paulo"
}
```

MONGO DB – REALIZANDO O TESTE DA APLICAÇÃO - ALTERAR

Teste Alterar

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:

Bairro: Estado: Cidade:



Preencha todos os campos com os valores diferentes do 1º teste, mantendo apenas o mesmo código (01).

```
_id: ObjectId('654ac5f5e05d8af1616e400b')  
código: "01"  
nome: "Teste"  
idade: 15  
endereço: "Outra"  
cpf: "3456"  
bairro: "Centro"  
cidade: "Belo Horizonte"  
estado: "Minas Gerais"
```



Atualizado no Mongo DB

MONGO DB – REALIZANDO O TESTE DA APLICAÇÃO - EXCLUIR

Teste Excluir

Cadastro de Clientes

Código:

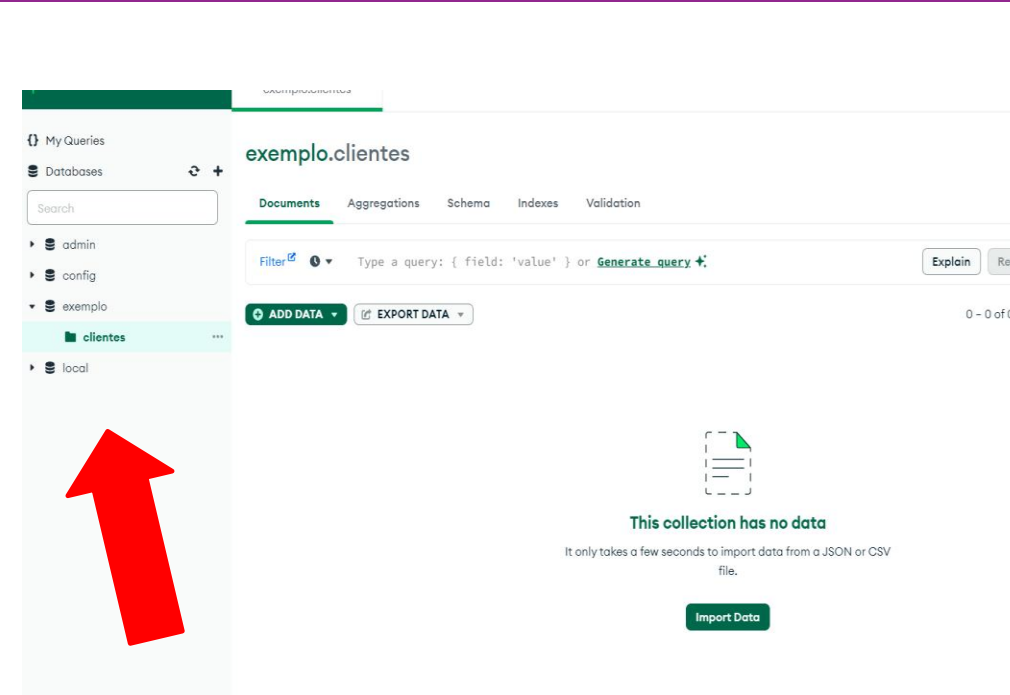
Nome: CPF:

Idade: Rua:

Bairro: Estado: Cidade:



Digite no campo código – 01
Click em Excluir



**Nenhuma Collection
Existente**

MONGO DB – CONSULTAR OS DADOS

No código existente adicionar a função

```
def consultar():
    codigo = txt_codigo.get()
    resultado = collection.find_one({"código": codigo})

    if resultado:
        txt_nome.insert(END, f" {resultado['nome']}")
        txt_idade.insert(END, f" {resultado['idade']}")
        txt_end.insert(END, f" {resultado['endereço']}")
        txt_cpf.insert(END, f" {resultado['cpf']}")
        txt_bairro.insert(END, f" {resultado['bairro']}")
        txt_cidade.insert(END, f" {resultado['cidade']}")
        comboestado.insert(END, f" {resultado['estado']}")
    else:
        lbl_resultado.config(text="Nenhum resultado encontrado")

lbl_resultado = Label(tela, text="", bg="#ffffff")
lbl_resultado.place(x=490, y=310)
```

MONGO DB – CONSULTAR OS DADOS

Explicação do Código

```
def consultar():  
    codigo = txt_codigo.get()
```

`codigo = txt_codigo.get()`: Esta linha obtém o valor digitado no campo de entrada `txt_codigo` e armazena-o na variável `codigo`.

O código é usado como critério de pesquisa para localizar um cliente no banco de dados.

```
def consultar():  
    codigo = txt_codigo.get()  
    resultado = collection.find_one({"código": codigo})
```

`resultado = collection.find_one({"código": codigo})`: Aqui, a função `find_one` é usada para buscar um documento no banco de dados MongoDB na coleção `collection`. O critério de pesquisa é o campo "código" no banco de dados, que é comparado ao valor armazenado na variável `codigo`. A função `find_one` retorna o primeiro documento que atende ao critério de pesquisa ou `None` se nenhum documento for encontrado.

MONGO DB – CONSULTAR OS DADOS

```
def consultar():  
    codigo = txt_codigo.get()  
    resultado = collection.find_one({"código": codigo})  
  
    if resultado:  
        txt_nome.insert(END, f" {resultado['nome']}\n")  
        txt_idade.insert(END, f"{resultado['idade']}\n")  
        txt_end.insert(END, f" {resultado['endereço']}\n")  
        txt_cpf.insert(END, f"{resultado['cpf']}\n")  
        txt_bairro.insert(END, f" {resultado['bairro']}\n")  
        txt_cidade.insert(END, f"{resultado['cidade']}\n")  
        comboestado.insert(END, f" {resultado['estado']}\n")  
    else:  
        lbl_resultado.config(text="Nenhum resultado encontrado")
```

Se um resultado for encontrado (if resultado), os detalhes do cliente são exibidos nos campos de entrada txt_nome, txt_idade, txt_end, txt_cpf, txt_bairro, txt_cidade, e no widget comboestado.

Caso nenhum resultado seja encontrado, a label lbl_resultado é atualizada com o texto "Nenhum resultado encontrado".

.

MONGO DB – CONSULTAR OS DADOS

Adicionar o botão consultar

```
btn_salvar = Button(tela, text="Salvar", width=10, command=salvar).place(x=130, y=280)
btn_alterar = Button(tela, text="Alterar", width=10, command=atualizar).place(x=220, y=280)
btn_excluir = Button(tela, text="Excluir", width=10, command=apagar).place(x=310, y=280)
btn_consultar = Button(tela, text="Consultar", width=10, command=consultar).place(x=490, y=280)
btn_sair = Button(tela, text="Sair", width=10, command=tela.quit).place(x=400, y=280)
tela.mainloop()
```

MONGO DB – REALIZANDO O TESTE DA APLICAÇÃO - CONSULTAR

Exemplo Mongo DB

Teste Consultar

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:

Bairro: Estado: Cidade:



Digite o campo código o valor existente no mongo db e depois click em consultar

Exemplo Mongo DB

Cadastro de Clientes

Código:

Nome: CPF:

Idade: Rua:


Bairro: Estado: Cidade:



Resultado

EXERCÍCIO PRÁTICO

Construa uma interface para cadastro de Pessoas no NO-SQL MONGO, como a tela



Código:


Nome: Idade:

Sexo: ☒ M ☐ F Altura: Peso: cidade:


Data Nascimento: Data Cadastro:

Data Atualização:


Escolher imagem




Salvar



Excluir




Alterar



Consultar

Sair



Codigo:1 Nome Luiz Claudio
Idade: 22 Sexo: m
Altura: 1,65 Peso: 50.0
Cidade: Registro Data Nasc: 10-11-2000



EXERCÍCIO PRÁTICO

Construa uma interface para cadastro de Veículos no NO-SQL MONGO, como a tela abaixo

Código: 21

Nome Veiculo: Mobi

Placa: PIA-24221

Utilitario:

☒ Utilitário
 ☐ Não utilitário


Salvar

Excluir

Alterar

Consultar

Sair



Escolher imagem

Modelo: Hatch


Marca: Chevrolet

Fiat
 Chevrolet
 Suzuki
 Volkswagen
 Renault
 Ford



EXERCÍCIO PRÁTICO

Construa uma interface para cadastro de Lugares Turísticos, no NO-SQL MONGO como a tela abaixo



Código: 12

Nome Local: Torre Eiffel





Valor Entrada: 10,00


Cidade: Paris

Estado: MG

Necessita Guia:

☒ Sim
 ☐ Não

 Salvar
  Excluir
  Alterar
  Consultar

 Sair





MAYLON.OLIVEIRA2@FATEC.SP.GOV.BR