senai- Cept rft

desenvolvimento de sistema

ana julia cunha, daniel azevedo, JOÃO MÁRCIO, Pablo aymar e VICTOR arthur

PROJETO FINAL

GERENCIAMENTO DE AGÊNCIA DE VIAGEM

SÃO LUÍS

2024

ANA JULIA CUNHA, DANIEL AZEVEDO, JOÃO MÁRCIO, PABLO AYMAR E VICTOR ARTHUR

PROJETO FINAL

GERENCIAMENTO DE AGÊNCIA DE VIAGEM

Trabalho apresentado a instituição com o requisito parcial para obtenção de notas em desenvolvimento de sistema.

Orientador: prof. Bruno Anderson

são luis

2024

sumário

**INTRODUÇÃO 4**

OBJETIVO5

TAREFAS DE CADA MEMBRO 6

# Introdução

Este trabalho traz informações sobre tudo que foi aprendido durante as aulas da UC Programação de Aplicativos, buscando abranger informações e estudo sobre o tema, como finalidade promover debates ao longo dos vários assuntos abordados através do conteúdo. As observações foram tiradas e estudas de matérias que expressam o assunto, aqui debatido e está divido em partes que se dividem em capa, introdução, desenvolvimento e conclusão.

# OBJETIVO.

O objetivo deste projeto é criar um sistema automatizado para uma agência de viagens, capaz de gerenciar pacotes de turismo de maneira eficiente. A aplicação, desenvolvida em Python, utilizará programação orientada a objetos e integração com um banco de dados, permitindo a inclusão, consulta, atualização e exclusão de pacotes de viagem. O sistema será projetado para facilitar o gerenciamento das informações dos pacotes, como destino, preço e descrição, garantindo uma organização clara e um processo de manutenção simples e funcional para a agência.

**Os Alunos Victor Arthur e Pablo Aymar, foram responsáveis pelo seguinte código;**

import sqlite3

class PacoteViagem:

def conexao(self):

# 1.estabelecendo conexão com oo banco de dados.

conexao = sqlite3.connect('agencia.db')

# 2.verificar se a tabela existe ou não, se não, criará.

tabela = '''

CREATE TABLE IF NOT EXISTS pacotes(

id INTEGER PRIMARY KEY AUTOINCREMENT,

destino VARCHAR(100),

preco FLOAT,

descricao VARCHAR(300)

);

'''

consulta = conexao.cursor()

consulta.execute(tabela)

return conexao

#“cadastrarPacote”: recebe como parâmetros o destino, preço e descrição de um pacote e insere um novo pacote no banco de dados.

def cadastrarPacote(self, destino, preco, descricao):

conexao = self.conexao()

consulta = conexao.cursor()

sql = 'INSERT INTO pacotes VALUES(?,?,?,?)'

campos = (None, destino, preco, descricao)

consulta.execute(sql, campos)

conexao.commit()

print('Pacote cadastrado com sucesso.')

conexao.close()

#“consultarPacotes”: recupera todos os pacotes cadastrados no banco de dados e imprime as informações de cada pacote (destino, preço e descrição).

def consultarPacotes(self):

conexao = self.conexao()

consulta = conexao.cursor()

sql = 'SELECT \* FROM pacotes'

consulta.execute(sql)

resultado = consulta.fetchall()

for itens in resultado:

print(f'Id: {itens[0]}, Destino: {itens[1]}')

conexao.close()

#“deletarPacote”: recebe como parâmetro o ID do pacote e, caso o pacote exista, deve ser mostrada uma mensagem de sucesso. Caso contrário, deve ser exibida uma mensagem de erro.

def deletarPacote(self, id):

conexao = self.conexao()

consulta = conexao.cursor()

sql = 'DELETE FROM pacotes WHERE id = ?'

campos = (id,)

consulta.execute(sql, campos)

conexao.commit()

if consulta.rowcount >= 1:

print('Deletado com sucesso.')

else:

print('Erro.')

import sqlite3

class Animal:

def conexao(self):

# 1.estabelecendo conexão com oo banco de dados.

conexao = sqlite3.connect('clinicaVeterinaria.db')

# 2.verificar se a tabela existe ou não, se não, criará.

tabela = '''

CREATE TABLE IF NOT EXISTS clinica(

codigo INTEGER PRIMARY KEY AUTOINCREMENT,

nome VARCHAR(100),

tipo\_animal VARCHAR(50),

idade INTEGER,

proprietario VARCHAR(100),

telefone\_contato INTEGER

);

'''

consulta = conexao.cursor()

consulta.execute(tabela)

return conexao

#cadastrarAnimal: Recebe como parâmetros o nome, tipo de animal, idade, proprietário e telefone de contato, e insere um novo animal no banco de dados.

def cadastrarAnimal(self, nome, tipo\_animal, idade, proprietario, telefone\_contato):

conexao = self.conexao()

consulta = conexao.cursor()

sql = 'INSERT INTO clinica VALUES(?,?,?,?,?,?)'

campos = (None, nome, tipo\_animal, idade, proprietario, telefone\_contato)

consulta.execute(sql, campos)

conexao.commit()

print('Animal cadastrado com sucesso.')

conexao.close()

def consultarAnimais(self):

conexao = self.conexao()

consulta = conexao.cursor()

sql = 'SELECT \* FROM clinica'

consulta.execute(sql)

resultado = consulta.fetchall()

for itens in resultado:

print(f'Código: {itens[0]}, Nome: {itens[1]}')

conexao.close()

def deletarAnimal(self, codigo):

conexao = self.conexao()

consulta = conexao.cursor()

sql = 'DELETE FROM clinica WHERE codigo = ?'

campos = (codigo,)

consulta.execute(sql, campos)

conexao.commit()

if consulta.rowcount >= 1:

print('Deletado com sucesso.')

else:

print('Erro.')

conexao.close()

def atualizarAnimal(self, codigo, change):

conexao = self.conexao()

consulta = conexao.cursor()

if change == 1:

nome = input('Informe um novo nome: ')

sql = "UPDATE clinica SET nome = ? WHERE codigo = ?"

campos = (nome, codigo)

consulta.execute(sql, campos)

elif change == 2:

tipo\_animal = input('Informe o tipo do animal: ')

sql = "UPDATE clinica SET tipo\_animal = ? WHERE codigo = ?"

campos = (tipo\_animal, codigo)

consulta.execute(sql, campos)

elif change == 3:

idade = input('Corrija a idade: ')

sql = "UPDATE clinica SET idade = ? WHERE codigo = ?"

campos = (idade, codigo)

consulta.execute(sql, campos)

elif change == 4:

proprietario = input('Informe o nome do proprietário: ')

sql = "UPDATE clinica SET proprietario = ? WHERE codigo = ?"

campos = (proprietario, codigo)

consulta.execute(sql, campos)

elif change == 5:

telefone\_contato = input('Informe o telefone para contato: ')

sql = "UPDATE clinica SET telefone\_contato = ? WHERE codigo = ?"

campos = (telefone\_contato, codigo)

consulta.execute(sql, campos)

conexao.commit()

conexao.close()

def consultarAnimalIndividual(self, codigo):

conexao = self.conexao()

consulta = conexao.cursor()

sql = 'SELECT \* FROM clinica WHERE codigo = ?'

campos = (codigo,)

consulta.execute(sql, campos)

resultado = consulta.fetchall()

print(resultado)

conexao.close()

#“atualizarPacote”: recebe como parâmetro o ID do pacote e alguma (ou toda) informação que se deseja atualizar. Fica a seu critério escolher quais dados deverão ser atualizados.

def atualizarPacote(self, id):

conexao = self.conexao()

consulta = conexao.cursor()

print(consulta.rowcount)

if consulta.rowcount <= 1:

print('-> Campos para mudança:\n1.Destino\n2.Preço\n3.Descricao')

change = int(input('Informe o campo escolhido: '))

if change == 'destino' or change == 'Destino' or change == 1:

destino = input('Informe o seu novo destino: ')

sql = 'UPDATE pacotes SET destino = ? WHERE id = ?'

campos = (destino, id)

consulta.execute(sql, campos)

elif change == 'preco' or change == 'Preco' or change == 2:

preco = input('Informe o novo preço: ')

sql = 'UPDATE pacotes SET preco = ? WHERE id = ?'

campos = (preco, id)

consulta.execute(sql, campos)

elif change == 'descricao' or change == 'Descricao' or change == 3:

descricao = input('Informe uma nova descrição do destino: ')

sql = 'UPDATE pacotes SET descricao = ? WHERE id = ?'

campos = (descricao, id)

consulta.execute(sql, campos)

else:

print('Erro.')

conexao.commit()

conexao.close()

#“consultarPacoteIndividual”: recebe como parâmetro o código do pacote e recupera apenas esse registro da tabela e exibe todas as informações apenas do item que possui esse ID, ou seja, será feita uma consulta no banco utilizando o código como filtro para retornar um único item da tabela.

def consultarPacoteIndividual(self, id):

conexao = self.conexao()

consulta = conexao.cursor()

if consulta.rowcount <= 1:

sql = 'SELECT \* FROM pacotes WHERE id = ?'

campos = (id,)

consulta.execute(sql, campos)

resultado = consulta.fetchall()

for itens in resultado:

print(f'Id: {itens[0]}, Destino: {itens[1]}, Preço: {itens[2]}, Descrição: {itens[3]}')

#for linha in consulta.fetchall():

# if id == linha[0]:

# print('Pacote: ',linha)

else:

print('Erro.')

conexao.commit()

conexao.close()

**Os alunos Daniel E João Márcio, foram responsáveis pelo código Principal;**

from pacoteViagem import PacoteViagem

opcao = 0

while opcao != 6:

print('--- Agência de Viagem ---')

print('1.Cadastrar Pacote;\n2.Consultar Pacotes;\n3.Deletar Pacote;\n4.Atualizar Pacote;\n5.Consultar Pacote Individual;\n6.Sair')

opcao = int(input('Selecione uma opção: '))

pacote = PacoteViagem()

if opcao == 1:

destino = str(input('Destino da Viagem: '))

preco = float(input('Preço do Pacote: '))

descricao = input('Descreva o destino: ')

pacote.cadastrarPacote(destino, preco, descricao)

print('===============')

elif opcao == 2:

print('===', 'Pacotes:')

pacote.consultarPacotes()

print('===============')

elif opcao == 3:

print('Pacotes:')

pacote.consultarPacotes()

id = int(input('Informe o Id do Pacote: '))

pacote.deletarPacote(id)

print('===============')

elif opcao == 4:

print('Pacotes:')

pacote.consultarPacotes()

id = int(input('Informe o Id do Pacote: '))

pacote.atualizarPacote(id)

print('===============')

elif opcao == 5:

print('Pacotes:')

pacote.consultarPacotes()

id = int(input('Informe o Id do Pacote: '))

pacote.consultarPacoteIndividual(id)

print('===============')

elif opcao == 6:

print('===============')

break

print('Seja Sempre Bem-Vindo(a)!')

**A aluna Ana Julia Foi Responsável pela elaboração do Trabalho Escrito/ Relatório- Todo o Trabalho.**

# conclusão

O desenvolvimento deste sistema de gerenciamento de pacotes de viagem permitiu a aplicação prática dos conceitos de programação orientada a objetos e manipulação de banco de dados em Python. A solução implementada oferece à agência de viagens uma ferramenta eficiente para gerenciar seus pacotes de forma organizada, permitindo cadastrar novos pacotes, consultar informações detalhadas, atualizar dados e excluir registros. O projeto não só atingiu seus objetivos técnicos, como também proporcionou uma experiência valiosa na integração de diferentes tecnologias e no desenvolvimento de software orientado a dados. Com isso, o sistema demonstra ser uma solução robusta e escalável para atender às necessidades de agências de turismo.