COMPUTATIONAL THINKING USING PYTHON

SPRINT 4

GUSTAVO IMPARATO CHAVES .....RM 551988

IZABELLY DE OLIVEIRA MENEZES.....RM 551423

JOÃO VITO SANTIAGO DA SILVA.....RM 86239

LUCAS MONTE VERDE.....RM 551604

MARCOS HENRIQUE GARRIDO DA SILVA.....RM 99578

SUMÁRIO

APRESENTAÇÃO

Em um mundo movido pela inovação, a excelência em projetos é fundamental para impulsionar avanços e melhorar as experiências cotidianas, visando aprimorar essa experiência, nós da **Chroma Cycle** apresentamos um projeto inovador de automatização da vistoria de bicicletas, utilizando inteligência artificial para tornar esse processo mais eficiente e econômico para a seguradora e seus clientes, o principal objetivo deste projeto é revolucionar a maneira como as vistorias de bicicletas para seguros são conduzidas.

Este empreendimento não é apenas uma resposta aos desafios atuais, mas promessa de eficiência, sustentabilidade e inovação, ao empregar tecnologias avançadas, nós pretendemos substituir as vistorias manuais por um sistema automatizado, garantindo não apenas precisão na avaliação, mas também a redução significativa de custo e de tempo para o nosso cliente e para seus clientes.

O nosso projeto vai contar com:

- Sistema de captura de imagens avançado: Onde implementaremos um sistema de captura de imagens de última resolução, capaz de registrar detalhes minuciosos da bicicleta, (esse componente é crucial para a análise abrangente das condições do veículo).

- Algoritmos de processamento de imagens onde utilizaremos algoritmos sofisticados de processamento de imagens para identificar e analisar características como arranhões, amassados, corrosões e outras irregularidade (esses algoritmos serão trinados para reconhecer padrões com base em conjuntos de dados abrangentes).

- Desenvolveremos um modelo de inteligência artificial capaz de interpretar as informações obtidas pelas imagens (este modelo será treinado para tomar decisões precisas sobre a elegibilidade da bicicleta para o seguro, considerando uma variedade de fatores).

- Disponibilizaremos uma interface de usuário intuitiva que permitirá que os usuários interajam, facilmente com o sistema, além de fornecer feedback

instantâneo sobre o processo da vistoria, a interface apresentará informações de maneira clara e compreensível.

O projeto traz diversos benefícios, como:

- A redução substancial de custos: A automação da vistoria elimina a necessidade de vistoriadores presenciais, reduzindo custos operacionais e, por consequência, tornando o seguro mais acessível para os clientes.

- A eficiência operacional: A inteligência artificial é capaz de analisar múltiplas bicicletas simultaneamente, resultando em uma reposta rápida e eficiente, isso não apenas economiza tempo, mas também agiliza todo o processo de obtenção do seguro.

- Experiência aprimorada para o cliente: Os clientes e futuros segurados desfrutarão de um processo de vistoria mais conveniente e menos demorado, incentivando a adesão ao seguro, isso fortalece a relação entre a seguradora e o cliente.

- Contribuição para a sustentabilidade: Ao reduzir a necessidade de deslocamento físico para vistorias, o projeto contribui indiretamente para práticas mais sustentáveis, alinhando-se com as crescentes preocupações ambientais.

O projeto da **Chroma Cycle** para a automatização de vistoria de bicicleta para o seguro é uma resposta inovadora aos desafios tradicionais associados a esse processo. Ao incorporar tecnologias de ponta, esperamos não apenas simplificar as vistorias, mas também estabelecer um novo padrão de eficiência econômica para a indústria de seguros de bicicletas. Este projeto representa não apenas uma mudança tecnológica, mas uma transformação na maneira como as seguradoras interagem com seus clientes, proporcionando benefícios significativos para ambas as partes, não é apenas uma resposta aos desafios atuais, mas uma promessa de um futuro mais eficiente e dinâmico com um compromisso contínuo com a excelência e a inovação, **Chroma Cycle** representa não apenas uma realização, mas um ponto de partida para novas conquistas e descobertas.

ESCOPO

TRELLO:

https://trello.com/invite/b/b0yJQL6y/ATTI293cf372c91487a121b92bd1de6716d82C716F96/chromacycle

PRINT CÓDIGO

```
import requests
from APIDatabase import *

def api_cep():
    cont = True
    while cont:
        cep = valida_num_str('Digite seu cep: ', 'cep')
        try:
            link = f'https://viacep.com.br/ws/{cep}/json/'
            requisicao = requests.get(link)
            dic_requisicao = requisicao.json()
            estado = dic_requisicao['uf']
            cidade = dic_requisicao['localidade']
            bairro = dic_requisicao['bairro']
            endereco = dic_requisicao['logradouro']
            cont = False
        except Exception as e:
            print(f"Erro na conexao: {e}")
    return cep, estado, cidade, bairro, endereco

def valida_input(info, tipo):
    val = True
    while val == True:
        match tipo:
            case 'int':
                try:
                    x = int(input(info))
                    val = False
                except:
                    print('Valor inválido')
            case 'float':
                try:
                    x = float(input(info))
                    val = False
                except:
                    print('Valor inválido')
            case 'str':
                try:
                    x = input(info)
                    val = False
                except:
                    print('Valor inválido')
    return x

def valida_num_str(info, tipo):
    valido = False
    str_cep = ('1', '2', '3', '4', '5', '6', '7', '8', '9', '0')
    str_cpf = ('1', '2', '3', '4', '5', '6', '7', '8', '9', '0', 'x', 'X')
    while not valido:
        dados = input(info)
        contagem = 0
        if tipo in 'cep' and len(dados) == 8:
            for i in dados:
```

```
                if i in str_cep:
                    contagem += 1
        elif tipo in 'cpf' and len(dados) == 11:
            for i in dados:
                if i in str_cpf:
                    contagem += 1
        else:
            print('Digite um número válido.')
        valido = True if contagem == len(dados) else None
    return dados

def retorna_dados(CPF):
    dados = {}
    dados["Cliente"]= select_cliente(CPF)
    dados["Bike"]= select_bike(CPF)
    dados["Acessorio"]= select_acessorio(dados.get('Numero_Serie'))
    return dados

def cadastro():
    print('----------------Cadastrando dados----------------\n')
    dados = {}
    pf = {}
    pf["Nome"] = valida_input('Digite seu Nome: ', 'str').title()
    pf["Email"] = valida_input('Digite seu E-mail: ', 'str')
    pf["Cpf"] = valida_num_str('Digite seu CPF: ', 'cpf')
    pf["Telefone"] = valida_input('Digite seu Telefone: ', 'str')
    pf["Cep"], pf["Estado"], pf["Cidade"], pf["Bairro"],pf["Endereço"] =
api_cep()
    pf["Numeracao"] = valida_input('Digite a numeração da residência: ',
'str')
    pf["Complemento"] = valida_input('Digite o complemento(caso haja): ',
'str')
    dados['Cliente'] = pf
    insert_cliente(dados['Cliente'])

#-------------------------------------------------------------------------
------------------------

def login(): #Função para logar o usuário
    print("------------Tela de Login------------\n")
    print("Digite 0 para retornar ao menu anterior\n")
    logado = False
    while logado == False:
        login = valida_input('Digite seu Email: ', 'str')
        senha = valida_input('Digite sua CPF: ', 'str')
        if select_login(login, senha) and login != '0' and senha != '0':
            logado = True
        elif login == '0' or senha == '0':
            break
        else:
            print('Email e(ou) senha incorreto(s)')
            logado = False
    return logado, senha

#-------------------------------------------------------------------------
------------------------
```

```
def altera_cadastro(CPF): #Função para alterar dados cadastrados do usuário
    print('----------------Alterando seu cadastro----------------\n')
    print('Para retornar ao menu anterior digite 0')
    dados = retorna_dados(CPF)
    mantem = True
    loop = ('SIM')
    sim = ('SIM, S')
    nao = ('NÃO, N, NAO')
    while mantem:
        y = 1
        for i in dados.keys():
            if 'Cliente' in i:
                print(f'Para alterar seu cadastro de informações pessoais
digite {y}')
            if 'Bike' in i and dados['Bike']:
                print(f'Para alterar seu cadastro da bike
{dados[i].get("Modelo")} digite {y}\n')
            y += 1
        opcao = valida_input('Digite a opção desejada: ', 'int')
        if opcao == 0:
            break
        i = 0
        y = 0
        for keys, value in dados.items():
            if i == opcao:
                for p1, p2 in value.items():
                    print(f'Para alterar {p1} cadastrado {p2} digite {y}')
                    y += 1
                opcao2 = valida_input('Digite a opção desejada: ', 'int')
                y = 0
                for p1, p2 in value.items():
                    if y == opcao2:
                        if type(dados[keys][p1]) == int:
                            att = valida_input(f'Para alterar
{dados[keys][p1]} Digite: ', 'int')
                            update(keys, p1, att, dados[keys].get('Cpf'))
                        elif type(dados[keys][p1]) == float:
                            att = valida_input(f'Para alterar
{dados[keys][p1]} Digite: ', 'float')
                            update(keys, p1, att, dados[keys].get('Cpf'))
                        else:
                            att = valida_input(f'Para alterar
{dados[keys][p1]} Digite: ', 'str')
                            update(keys, p1, att, dados[keys].get('Cpf'))
                    y += 1
            i += 1
        loop = valida_input("deseja alterar mais alguma informação em seu
cadastro ? Digite sim ou não.", 'str')
        loop = loop.upper()
        if loop in sim:
            mantem = True
        elif loop in nao:
            mantem = False
    return dados
```

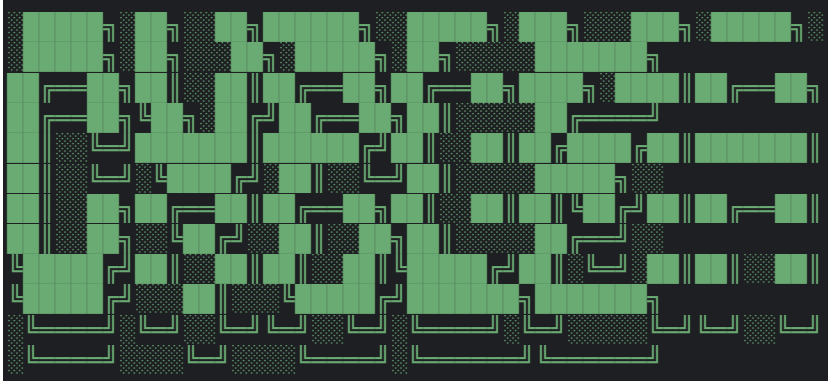
```
#-------------------------------------------------------------------------------
------------------------

def add_bike(CPF): #Função para adicionar/cadastrar bikes
    print('----------------Cadastrando Bikes----------------\n')
    for i in range(0, valida_input('Digite a quantidade de bikes: ', 'int')):
        bike = {}
        acessorio = {}
        bike["Modelo"] = valida_input(f'Digite o modelo da {i + 1}ª bike: ',
'str')
        bike["Numero_Serie"] = valida_input(f'Digite o número de série da {i
+ 1}ª bike: ', 'str')
        bike["Lancamento"] = valida_input('Digite o ano de lançamento da
bike: ', 'int')
        bike["Valor"] = valida_input(f'Digite o valor da {i + 1}ª bike: R$ ',
'float')
        insert_bike(bike, CPF)
        for acess in range(0, valida_input('Digite a quantidade de
acessórios: ', 'int')):
            acessorio["Acessório"] = valida_input(f'Digite qual o {acess +
1}° acessório instalado na bike: ', 'str')
            acessorio["Valor"] = valida_input(f'Digite o valor do {acess +
1}° acessório: R$ ', 'float')
            insert_acessorio(acessorio, bike["Nr_Serie_Bike"])



#-------------------------------------------------------------------------------
------------------------

def remove_bike(CPF):
    mantem = True
    sim = {'SIM', 'S'}
    nao = {'NÃO', 'N', 'NAO'}
    while mantem:
        print('----------------Removendo Bikes----------------\n')
        dados = {
            "Bike": select_bike(CPF),
            "Acessorio":
select_acessorio(select_bike(CPF).get('Numero_Serie'))
        }
        if dados['Bike'].get('Numero_Serie') is not None:
            for i, (key, value) in enumerate(dados.items()):
                if 'Bike' in key and key is not None:
                    print(f'Digite {i} para deletar seu cadastro da bike
{value.get("Modelo")}')
                    print(f'Com o número de série de
{value.get("Numero_Serie")}')
            opcao = valida_input('Digite a opção desejada: ', 'int')
            for i, (key, value) in enumerate(dados.items()):
                if i == opcao:
                    delete(value.get('Numero_Serie'))
                    dados.pop(key)
                    break
        else:
            print('\nVocê não tem nenhuma bike cadastrada!')
```

```
            break
        loop = valida_input('Deseja remover mais alguma bike? Digite SIM ou
NÃO: ', 'str').upper()
        if loop in sim:
            mantem = True
        elif loop in nao:
            mantem = False
        else:
            print('Opção inválida. Encerrando remoção de bikes.')
            mantem = False

#-------------------------------------------------------------------------
------------------------

def org_cadastro(dados): #Organizar dados após remover um item
    infos = {}
    infos['Cliente'] = dados['Cliente']
    index = 1
    for i in dados.keys():
        if 'Bike' in i:
            if not str(index) in i:
                bike = {}
                bike = dados['Bike'+str(index+1)]
                infos["Bike"+ str(index)] = bike
            else:
                bike = {}
                bike = dados['Bike'+str(index)]
                infos["Bike"+ str(index)] = bike
            index += 1
    return infos

#-------------------------------------------------------------------------
------------------------

def imprimindo_dados(CPF):
    print('---------------Dados Cadastrados---------------\n')
    dados = retorna_dados(CPF)
    print(dados)
    valor_total_bike = 0
    for i in dados:
        valor_por_bike = 0
        item = dados[i]
        print('\n************************************')
        if item is not None:
            for x, y in item.items():
                if item.get(x) is not None:# Remover
                    if 'Cliente' in i:
                        if x == 'cpf': #verificar o nome CPF e outros
                            print(f'{x} = {(y[0:3])}.{(y[3:6])}.{(y[6:9])}-
{(y[9:11])}\n')
                        elif x == 'Cep':
                            print(f'{x} = {(y[0:3])}.{(y[3:5])}-
{(y[5:8])}\n')
                        else:
                            print(f'{x} = {y}\n')
                    elif 'Bike' in i or 'Acessorio' in i:
```

```
                        if 'Vl' in x:
                            print(f'{x} = R$ {float(y):.2f}\n')
                            valor_total_bike += float(y)
                            valor_por_bike += float(y)
                        else:
                            print(f'{x} = {y}\n') if x != 'Cpf' else None
                    else:
                        print(f'{x} = {y}\n')
                elif 'Bike' in i:
                    print(f'Valor total da bike {dados[i].get("Modelo")} = R$
{valor_por_bike:.2f}') if 'Bike' in i else None
                    sinistro(dados[i].get('Lancamento'), valor_por_bike)
    print('-------------------------------------------')
    print(f'Valor total das bikes cadastradas R$: {valor_total_bike:.2f}')
    print('-------------------------------------------\n')

#-------------------------------------------------------------------------------
------------------------

def calc_anos(anos_corridos):
    match anos_corridos:
                case 0 | 1:
                    peso_ano = 0
                case 2 | 3 :
                    peso_ano = 1
                case 4 | 5 | 8:
                    peso_ano = 2
                case 6 | 7 | 9:
                    peso_ano = 2
                case _:
                    peso_ano = 3
    return peso_ano

#-------------------------------------------------------------------------------
------------------------

def calc_valor(valor_total):
        if valor_total < 5000:
            peso_valor = 0
        elif valor_total < 10000:
            peso_valor = 1
        elif valor_total < 15000:
            peso_valor = 2
        elif valor_total < 20000:
            peso_valor = 3
        else:
            peso_valor = 4
        return peso_valor

#-------------------------------------------------------------------------------
------------------------

def sinistro(ano,valor):
    ano_atual = 2023
    peso_ano = calc_anos(ano - ano_atual)
    peso_valor = calc_valor(valor)
```

```
    valor_da_escala = peso_ano + peso_valor
    if valor_da_escala > 10:
            print('Sua bicicleta não está elegível para contratar a
seguradora, sinto muito!', end='')
    else:
        match valor_da_escala:
            case 1 | 2 | 3:
                categoria = 'básico'
                custo = 0.06
            case 4 | 5 | 6:
                categoria = 'normal'
                custo = 0.07
            case 7 | 8:
                categoria = 'moderado'
                custo = 0.08
            case _:
                categoria = 'avançado'
                custo = 0.09
        print(f'Sua bicicleta está elegível para o seguro! Ele está na
categoria {categoria} e '
            f'seu custo é de R${(valor)*custo:.2f} anuais ou
R${((valor)*custo)/12:.2f} por mês.')

#-------------------------------------------------------------------------
------------------------

def menu(): #Menu principal do sistema
    create_table_cliente() if not verif_tabela('cliente') else None
    create_table_bike() if not verif_tabela('bike') else None
    create_table_acessorio() if not verif_tabela('acessorio') else None
    print('------------Menu do Sistema------------\n')
    opcao = -1
    logado = False
    while opcao != 0: #Loop para encerrar o sistema
        opcao = valida_input('\nDigite [1] para efetuar seu cadastro.
\nDigite [2] para fazer login. \nDigite [0] para encerrar o sistema.
\n\nDigite uma opção: ', 'int')
        match opcao:
            case 0: #Encerra o sistema
                pass
            case 1: #Caso parar cadastrar os dados
                cadastro()
            case 2: #Caso para fazer o login
                logado, cpf = login()
            case _:
                print('\n=======================================\n||
Digite um número válido      ||\n=======================================')

        while logado == True:
            dados = select_cliente(cpf)
            print('----------------Tela de Login----------------')
            print(f'----------------Bem Vindo(a) {dados.get("Nome")}---------
-------')
            opcao = valida_input('Digite [1] para alterar informações no seu
cadastro. \nDigite [2] para ver os dados do seu cadastro. \nDigite [3] para
adicionar bikes ao seu cadastro \nDigite [4] para remover bikes do seu
```

```
cadastro. \nDigite [0] para encerrar o sistema. \n\nDigite uma opção: ',
'int')
            match opcao:
                case 0: #Encerra o sistema
                    logado = False
                case 1:
                    altera_cadastro(cpf)
                case 2:
                    imprimindo_dados(cpf)
                case 3:
                    add_bike(cpf)
                case 4:
                    remove_bike(cpf)
                case _:
                    print('Digite um número válido')
#-------------------------------------------------------------------------------
-----------------------

def main():

    print('''
================================================================================
=================
================================================================================
==================
```



```
================================================================================
=================
================================================================================
=================
''')

    print('=-=-=-=Bem-vindo ao sistema da Chroma Cycle=-=-=-=\n')

    menu()

    print('\n\n=-=-=-=Encerrando programa!=-=-=-=\n')

#-------------------------------------------------------------------------------
------------------------

# M A I N
main()
```

VÍDEO YOUTUBE

YOUTUBE:

https://youtu.be/o3bm8bDEyw4?si=g8l4lzVZJAhr7SsT