

TP2 - Projeto de bloco



Esse é o segundo teste de performance da disciplina Projeto de Bloco.



Aluno: Bruno Fernandes

Email: bruno.fernandes@al.infnet.edu.br


Prof: Prof. Alcione Dolavale

Relatório:

Para acessar o repositório deste trabalho clique abaixo:

Bwendel26/PB_INFNET_Python_redes_SO

Um software cliente-servidor em Python que explore conceitos de arquitetura de redes, arquitetura de computadores e/ou de sistemas operacionais, acompanhado de relatório explicativo. Este

 https://github.com/Bwendel26/PB_INFNET_Python_redes_SO/tree/master/tp2



No TP2 o objetivo foi montar um pequeno sistema desktop de monitoramento dos componentes principais do computador. Foi codificado uma série de algoritmos, na linguagem Python, que utilizaram bibliotecas da linguagem para monitoramento do computador e para criar a interface que mostra os gráficos e informações a serem apresentadas pelo usuário.

Ferramentas utilizadas:

Linguagem de programação: Python (versão 3.8)

IDE: PyCharm Community Edition 64 bits

Bibliotecas utilizadas:

- Pygame

Pygame Front Page - pygame v2.0.0.dev25 documentation

Basic information about pygame: what it is, who is involved, and where to find it. Steps needed to compile pygame on several platforms. Also help on finding and installing prebuilt binaries for your


 <https://www.pygame.org/docs/>



- psutil

psutil documentation - psutil 5.7.4 documentation

psutil (python system and process utilities) is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors) in Python. It is useful mainly for system monitoring, profiling, limiting process resources and the management of running

 <https://psutil.readthedocs.io/en/latest/>

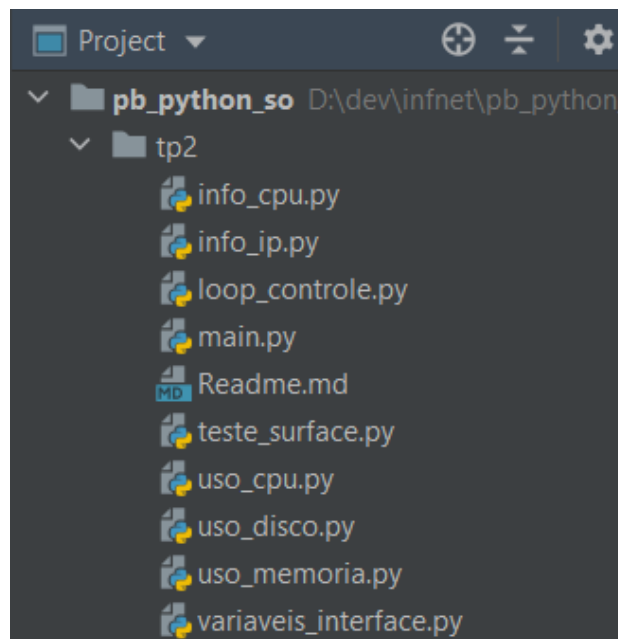
- platform

platform - Access to underlying platform's identifying data - Python 3.9.0 documentation

Source code: Lib/platform.py Note Specific platforms listed alphabetically, with Linux included in the Unix section. Queries the given executable (defaults to the Python interpreter binary) for various architecture information. Returns a tuple which contain information about the bit architecture and the

 <https://docs.python.org/3/library/platform.html>

A estrutura do projeto do TP2 foi a seguinte:



Como podemos ver os arquivos foram divididos um para cada funcionalidade, e o arquivo main.py (principal) chama todas as funções que executam o programa por um todo. Como podemos ver temos funções para cada chamada (uso_cpu.py, uso_memoria.py, uso_disco, info_cpu, etc...).

A estrutura do código e utilização de biblioteca pode ser vista no seguinte código:

```

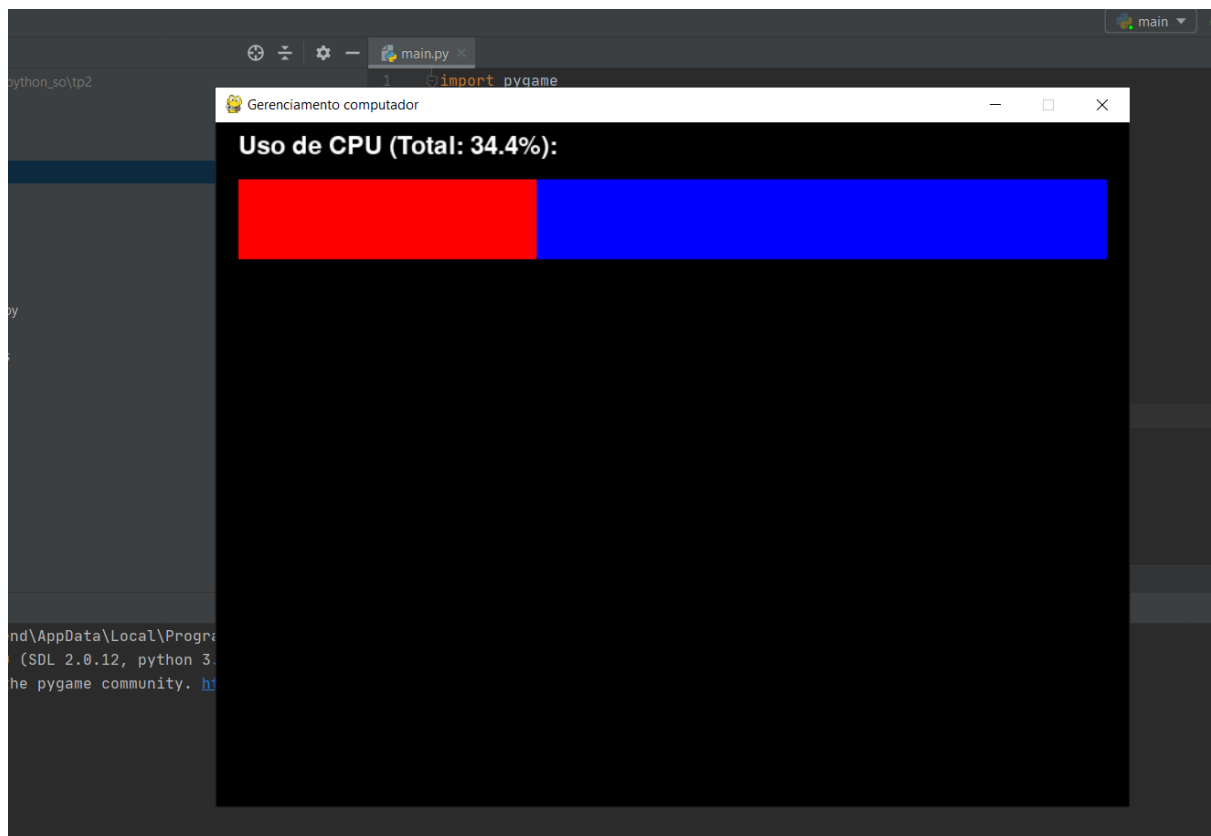
1 import psutil
2 import pygame
3 import variaveis_interface as int_vars
4
5 #funcs
6 def percentual_cpu():
7     """
8     Função que retorna o percentual de uso da CPU
9     do computador.
10    :param tempo: em segundos para retornar o percentual.
11    :return: float percentual_usado
12    """
13    percentual_usado = psutil.cpu_percent(interval=0)
14
15    return percentual_usado
16
17
18 surface1 = int_vars.s1
19
20 def mostra_uso_cpu():
21     capacidade = percentual_cpu()
22     larg = int_vars.tela_largura - 2*20
23     int_vars.tela.fill(int_vars.PRETO)
24     pygame.draw.rect(surface1, int_vars.AZUL, (20, 50, larg, 70))
25     larg = larg*capacidade/100
26     pygame.draw.rect(surface1, int_vars.VERMELHO, (20, 50, larg, 70))
27     texto_barra = "Uso de CPU (Total: " + str(capacidade) + "%):"
28     text = int_vars.font.render(texto_barra, 1, int_vars.BRANCO)
29     int_vars.tela.blit(surface1, (0, 0)) # setando divisao tela
30     int_vars.tela.blit(text, (20, 10))

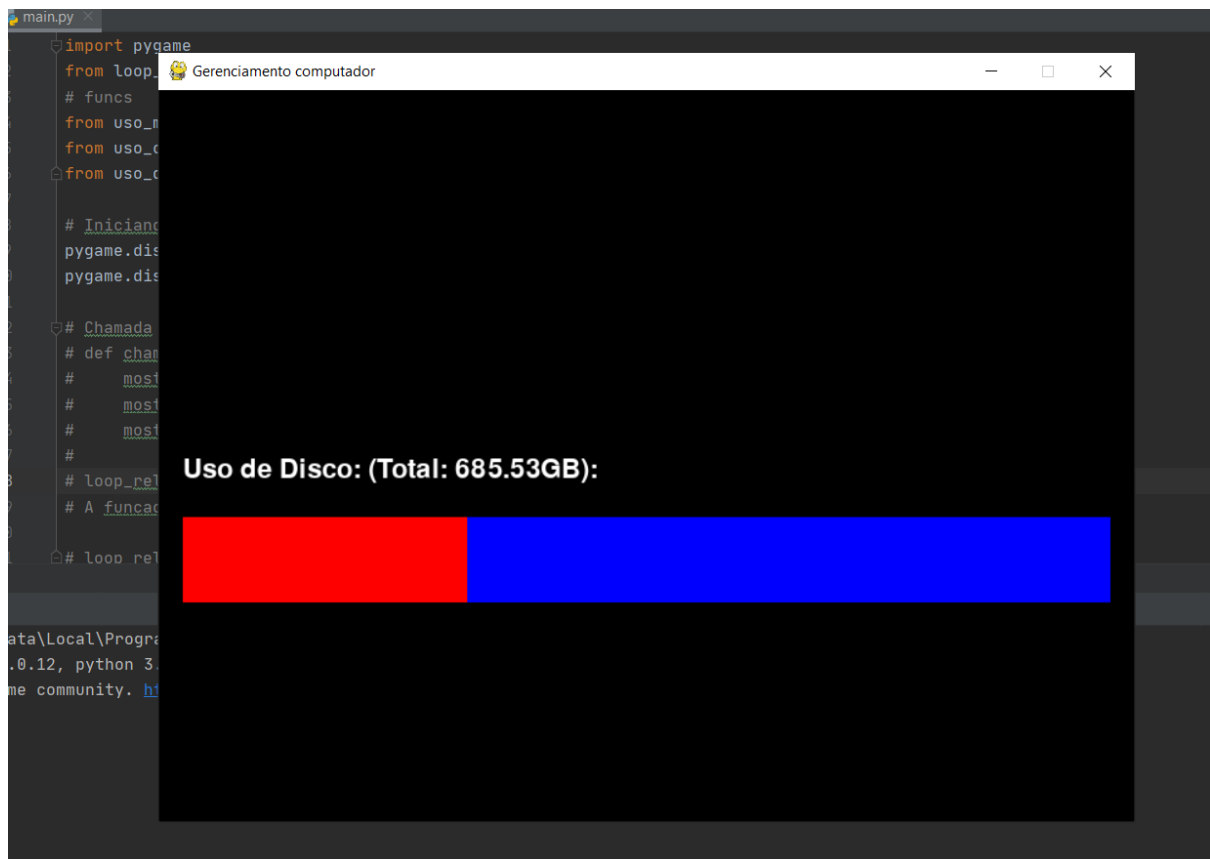
```

Como podemos ver é feito o import das bibliotecas necessárias e do arquivo que contém as variáveis gerais do projeto, logo em seguida iniciamos montando a função que utiliza a biblioteca psutil para pegar as informações do componente analisado, nesse exemplo é a CPU e posteriormente retorna o resultado.

No final o código retorna a apresentação de cada tela, porém o código indicando as surfaces no roteiro de aprendizagem

segue os prints:





E ao final temos os seguintes códigos que mostram informações de IP e sobre arquitetura da CPU:

IP:

The image shows a Python IDE with a file explorer on the left and a code editor on the right. The file explorer lists several files, with `info_ip.py` selected. The code editor displays the following Python code:

```
1 import psutil
2
3 #funções
4 def informacao_ip():
5     """
6     Função que retorna o endereço de IP
7     de um computador.
8     :return ip:
9     """
10
11     dic_interface = psutil.net_if_addrs()
12     ip = dic_interface["Ethernet"][1].address
13
14     return ip
15
16
17 print(informacao_ip())
```

Below the code editor, a terminal window shows the execution of the script:

```
C:\Users\bwend\AppData\Local\Programs\Python\Python38-32\python.exe D:/dev/infnet
169.254.87.195

Process finished with exit code 0
```

Informação da CPU:

The image shows a Python IDE with a file explorer on the left and a code editor in the center. The file explorer lists several files, with `info_cpu.py` selected. The code editor displays the following Python code:

```
4 #funções
5 def informacao_cpu():
6     """
7     Função que retorna as informações da CPU de
8     um computador.
9     :return informação:
10    """
11    num_cores = psutil.cpu_count()
12    num_cores_reais = psutil.cpu_count(logical=False) #numero de processadores s/
13    processador_bits = platform.processor()
14    processador_nome = platform.node()
15    processador_detalhes = platform.platform()
16    sistema_operacional = platform.system()
17
18    informacao = "Processador:\nNome computador: {}\nN° de cores: {}.\nN° de cores
19                \nArquitetura de {}\nSistema Operacional: {}\nDetalhes: {}" \
20                .format(processador_nome, num_cores, num_cores_reais, processad
21
22    return informacao
23
24 print(informacao_cpu())
25
```

Below the code editor, the output of the script is displayed in a terminal window:

```
Nome computador: DESKTOP-1HRM04L
N° de cores: 8.
N° de cores reais: 4.
Arquitetura de Intel64 Family 6 Model 158 Stepping 10, GenuineIntel
Sistema Operacional: Windows
Detalhes: Windows-10-10.0.19041-SP0

Process finished with exit code 0
```