INSTITUTO INFNET

CARLOS HENRIQUE DOS SANTOS RODRIGUES JUNIOR

ARQUITETURA DE COMPUTADORES, SISTEMAS OPERACIONAIS E REDES

PROJETO DE BLOCO – TP9

Rio de Janeiro, novembro

2020

CARLOS HENRIQUE DOS SANTOS RODRIGUES JUNIOR

ARQUITETURA DE COMPUTADORES, SISTEMAS OPERACIONAIS E REDES

PROJETO DE BLOCO – TP9

Trabalho apresentado no curso de graduação do Instituto Infnet.

Professor: Adriano Saad

Rio de Janeiro, novembro

2020

Sumário

[Capítulo 1 6](#_Toc57836055)

[Objetivo 6](#_Toc57836056)

[Bibliotecas utilizadas 6](#_Toc57836057)

[Desenvolvimento 6](#_Toc57836058)

[Problemas observados: 7](#_Toc57836059)

[Soluções de problemas anteriores: 7](#_Toc57836060)

[Capítulo 2 8](#_Toc57836061)

[Objetivo 8](#_Toc57836062)

[Bibliotecas utlizadas 8](#_Toc57836063)

[Desenvolvimento 8](#_Toc57836064)

[Problemas observados 10](#_Toc57836065)

[Soluções de problemas anteriores 10](#_Toc57836066)

[Capítulo 3 11](#_Toc57836067)

[Objetivo 11](#_Toc57836068)

[Bibliotecas utilizadas 11](#_Toc57836069)

[Desenvolvimento 11](#_Toc57836070)

[Problemas observados: 12](#_Toc57836071)

[Soluções de problemas anteriores: 12](#_Toc57836072)

[Correção proporção do tamanho do gráfico de Disco 12](#_Toc57836073)

[Correção proporção do tamanho do gráfico de Memória 12](#_Toc57836074)

[Correção carrossel 12](#_Toc57836075)

[Capítulo 4 14](#_Toc57836076)

[Objetivo 14](#_Toc57836077)

[Bibliotecas utilizadas 14](#_Toc57836078)

[Desenvolvimento 14](#_Toc57836079)

[Problemas observados: 15](#_Toc57836080)

[Soluções de problemas anteriores: 15](#_Toc57836081)

[Capítulo 5 16](#_Toc57836082)

[Objetivo 16](#_Toc57836083)

[Bibliotecas utilizadas 16](#_Toc57836084)

[Desenvolvimento 16](#_Toc57836085)

[Infraestrutura 16](#_Toc57836086)

[Problemas observados 16](#_Toc57836087)

[Soluções de problemas anteriores 17](#_Toc57836088)

[Capítulo 6 18](#_Toc57836089)

[Objetivo 18](#_Toc57836090)

[Bibliotecas 18](#_Toc57836091)

[Desenvolvimento 18](#_Toc57836092)

[Problemas observados: 19](#_Toc57836093)

[Soluções de problemas anteriores: 19](#_Toc57836094)

[Capítulo 7 20](#_Toc57836095)

[Objetivo 20](#_Toc57836096)

[Bibliotecas utilizadas 20](#_Toc57836097)

[Cliente 20](#_Toc57836098)

[Servidor 20](#_Toc57836099)

[Desenvolvimento 20](#_Toc57836100)

[Problemas observados 22](#_Toc57836101)

[Soluções de problemas anteriores 22](#_Toc57836102)

[Capítulo 8 23](#_Toc57836103)

[Análise xpto 23](#_Toc57836104)

[Análise bcd 23](#_Toc57836105)

[Análise bcd 23](#_Toc57836106)

[Análise bcd 23](#_Toc57836107)

[Anexos 24](#_Toc57836108)

[Código fonte v1 24](#_Toc57836109)

[Código fonte v2 30](#_Toc57836110)

[Código fonte v3 37](#_Toc57836111)

[Código fonte v4 46](#_Toc57836112)

[Código fonte v5 57](#_Toc57836113)

[Código fonte v6 69](#_Toc57836114)

[Código fonte v7 81](#_Toc57836115)

[Cliente: 81](#_Toc57836116)

[servidor: 100](#_Toc57836117)

[Telas 115](#_Toc57836118)

[Evidência v1 116](#_Toc57836119)

[Evidências v2 119](#_Toc57836120)

[Evidências v3 122](#_Toc57836121)

[Evidências v6 126](#_Toc57836122)

[Evidências v5 130](#_Toc57836123)

[Evidências v6: 133](#_Toc57836124)

[Evidência v7: 136](#_Toc57836125)

# Capítulo 1

Esta sessão contempla todas as bibliotecas e funções utilizadas para atender aos requisitos da versão 1 do projeto de bloco.

## Objetivo

* Criar uma barra de indicação da porcentagem do uso de memória
* Criar uma barra de indicação da porcentagem do uso de CPU
* Criar uma barra de indicação da porcentagem de uso do Disco
* Informar o IP da máquina

## Bibliotecas utilizadas

* Pygame
* Psutil
* Cpuinfo
* Platform

## Desenvolvimento

A aplicação entregue, consiste em um utilitário onde é possível visualizar os dados de IP da rede, uso da CPU e quantidade de memória utilizada.

Para se obter o IP (Internet Protocol) da máquina, foi criado uma função chamada getIp. Essa função precisa do pacote ***plataform*** e ***psutil*** instalado. O pacote ***plataform*** se faz necessário para se obter o sistema operacional em que o usuário está executando a aplicação para que concomitante a função ***net\_if\_addrs()*** da biblioteca ***psutil*** seja possível obter o IP. A função ***net\_if\_addrs()*** do pacote ***psutil*** retorna informações diferentes baseado no sistema operacional em execução. Dessa forma faz se necessário saber qual chave utilizar obter a informação correta.

Para obter o percentual de uso da CPU foi utilizado a função ***cpu\_percent(interval=0)***. Essa função retorna um ***float*** representando a utilização do sistema, ela pode receber dois parâmetros: ***interval*** e ***percpu***. ***Interval*** configura a velocidade em que os dados serão atualizados. Já o ***percpu*** recebe um booleano e quando igual a True, retorna um vetor contendo floats referente a utilização do sistema no tempo.

Afim de obter os dados da memória, foi utilizado a função ***virtual\_memory()***, que retorna dados estatísticos da memória do computador. Essa função retorna um objteto contendo os seguintes atributos: ***total*** e ***available***.

* Total: representa o espaço físico total
* Available: representa o espaço disponível não incluindo o swap

Por fim, para se obter os dados de disco, foi utilizado a função ***disk\_usage(‘/’)*** também do pacote ***psutil***. Essa função retorna as seguintes propriedades: total, used, free e percent.

* Total: representa o volume total da partição
* Used: representa o valor total usado pela partição
* Free: representa o espaço livre da partição
* Percent: representa o percentual usado pela partição

## Problemas observados:

Sem problemas observados.

## Soluções de problemas anteriores:

Sem soluções anteriores.

# Capítulo 2

Esta sessão contempla todas as bibliotecas e funções utilizadas para atender aos requisitos da versão 2 de projeto de bloco.

## Objetivo

* Extrair o código anterior e criar 5 visualizações diferentes:
  + Tela com informações do processador
    - Informar o modelo/nome da CPU
    - Informar o tipo de arquitetura
    - Adicionar informação da palavra do processador
    - Informar a frequência total e frequência de uso da CPU
    - Informar o número total de núcleos (físico) e threads (lógico)
  + Tela com informações de memória
  + Tela com informações de disco
  + Tela com informações de IP
* Implementar uma navegação em slide. Sempre que o usuário clicar nas setas da direita ou esquerda a tela deve mudar como um carrossel.
* Implementar uma tela de resumo

## Bibliotecas utlizadas

* Pygame
* Psutil
* Cpuinfo
* Platform

## Desenvolvimento

Nessa etapa, foi solicitado que os dados coletados no capítulo anterior, fossem apresentados em telas diferentes. Para isso, foram criadas 8 superfícies:

* Superficie\_info\_cpu
  + Superficie\_grafico\_cpu
* Superficie\_info\_disco
  + Superficie\_grafico\_disco
* Superficie\_info\_memoria
  + Superficie\_grafico\_memoria
* Superficie\_info\_rede
* Superficie\_resumo

Onde a superfície com o prefixo “info” seria responsável por exibir em tela os dados e as superfícies com o prefixo “gráfico” seriam responsáveis pela apresentação dos gráficos respectivos. Para alcançar tal objetivo, foi utilizado o módulo do ***pygame*** e a função ***Suface((largura\_tela, altura\_tela))***. As dimensões definidas para esse projeto foram 800x600, dessa forma todas as superfícies de informação a altura definida foi 600/3 ou senha 200 pixels.

Foi solicitado também que fosse implementado um carrossel para a troca de tela. Sempre que o usuário clicar na seta da direita ou esquerda a aplicação deve reagir a essa interação apresentando a nova tela. Para alcançar esse objetivo, foi criado uma variável auxiliar chamada de “***posicao\_atual***”. Durante o processo de renderização do pygame é feito um monitoramento dos eventos gerados pelo usuário, sempre que o evento “***pygame\_.KEYDOWN***” e “***pygame.K\_RIGHT***” ou “***pygame.K\_LEFT***” ou “***pygame.K\_SPACE***” for identificado é incrementado ou decrementado o valor 1 a variável “***posicao\_atual***”. Essa variável é passada pra função “***getEnvolucro(posicao\_atual)***” que é responsável por renderizar as superfícies respectivas.

|  |  |
| --- | --- |
| Posição atual | Superfície |
| 0 | Envolucro\_dados\_cpu() |
| 1 | Envolucro\_dados\_memoria() |
| 2 | Envolucro\_dados\_disco() |
| 3 | Envolucro\_dados\_rede() |
| 4 | Resumo() |

Para obter os dados de CPU, foi utilizado a biblioteca ***“cpuinfo”*** e a função “***get\_cpu\_info()”***. Onde o retorno obtido foi:

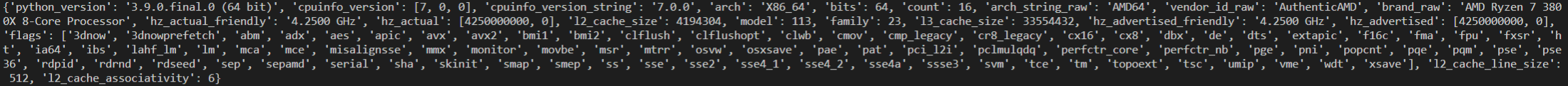


Figura Resposta CpuInfo

Para se obter as informações exigidas nesse capitulo, foi necessário obter os seguintes valores da chave do “***cpuinfo***”:

* Brand\_raw = Nome Processador
* Arch = Arquitetura do processador
* Bits = Bits do processador
* Hz\_actual\_friendly = Frequência de atuação do processador
* Count = Núcleos físicos do processador

Para se obter os dados do disco, também foi utilizado a biblioteca “***psutil***” com a função “***disk\_usage(‘/’)”***  onde a “**/**” informada como parâmetro faz referência ao disco principal da máquina.O resultado obtido foi:

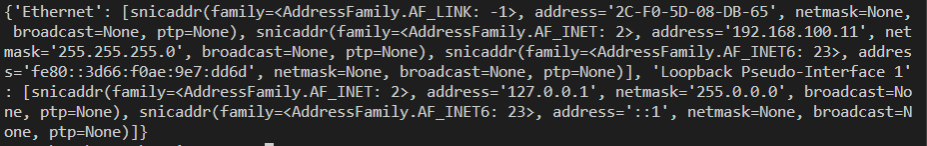


Para se obter as informações exigidas nesse capitulo, foi necessário obter os seguintes valores da chave do “***disk\_usage***”:

* Percent = Percentual usado
* Used = Kb usado
* Free = Kb livre

Para se obter o valor usado e livre em GB foi aplicado o seguinte cálculo:

Para se obter o endereço IP da máquina, também foi utilizado a biblioteca “***psutil***” e a função “***net\_if\_addrs()”.*** O resultado obtido pela função “***net\_if\_addrs()***”, foi:



Entretanto para se obter o endereço IP da máquina do usuário, faz se necessário saber em qual sistema operacional a aplicação está executando, pois a chave presente no JSON pode variar entre:

* Wlp3s0 para Linux
* Ethernet para Windows

## Problemas observados

* A proporção do tamanho das barras da memória e disco (HD) estão desproporcionais
* Ao chegar a última tela, o carrossel não retorna para a primeira página

## Soluções de problemas anteriores

Sem soluções anteriores.

# Capítulo 3

Esta sessão contempla todas as bibliotecas e funções utilizadas para atender aos requisitos da versão 3 de projeto de bloco.

## Objetivo

* Criar uma ou mais funções que retornem ou apresentem informações sobre diretórios e arquivos
* Criar uma ou mais funções que retornem ou apresentem informações sobre processos do sistema. As informações podem ser: PID, nome do executável, consumo de processamento, consumo de memória, entre outras

## Bibliotecas utilizadas

* Pygame
* Psutil
* Cpuinfo
* Subprocess
* Os
* Time
* Socket

## Desenvolvimento

Para atender ao requisito de obtenção dos dados de arquivos e diretórios, foi criado uma função chamada “***mostrar\_dados\_diretorio()***”, essa função retorna uma lista contendo os arquivos e diretórios presentes no path do projeto. Ela utiliza a biblioteca “***os***” e a função “***listdir()***”. O resultado obtido pela função foi:



Para se obter os detalhes dos documentos também foi utilizado a biblioteca “***os***”. Para se obter os valores solicitados, foram utilizadas as seguintes funções:

* St\_size: obter informações do tamanho do arquivo.
* St\_ctime: em alguns sistemas (como Unix) é a hora da última mudança de metadados e, em outros (como Windows), é a hora de criação.
* St\_mtime: hora da última modificação.

Por fim, para se obter os dados do processo, foi utilizado a biblioteca “***subprocess***” e a função “**popen().pid**”. Essa função inicia um processo e retorna o PID do mesmo. PIP é o código identificado do processo e é gerenciado pelo sistema operacional.

Para se obter os dados desejados do processo, fez se necessário obter os seguintes atributos:

* Memory\_percent
* Memory\_info.rss
* Create\_time()

## Problemas observados:

* A função responsável por obter o endereço IP da máquina do usuário, só está retornando o IP da posição inicial da tupla. Caso o usuário possua duas ou mais interfaces a função não obtém os demais endereços.

## Soluções de problemas anteriores:

### Correção proporção do tamanho do gráfico de Disco

O novo código para apresentar o gráfico do consumo do disco foi alterado para:

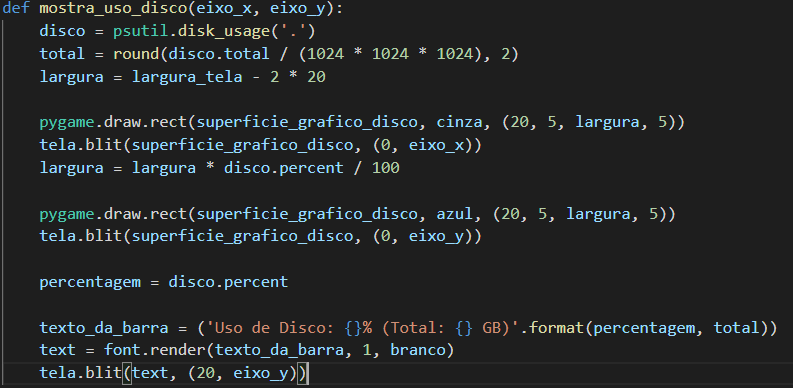


Figura Código para apresentar o consumo do disco

### Correção proporção do tamanho do gráfico de Memória

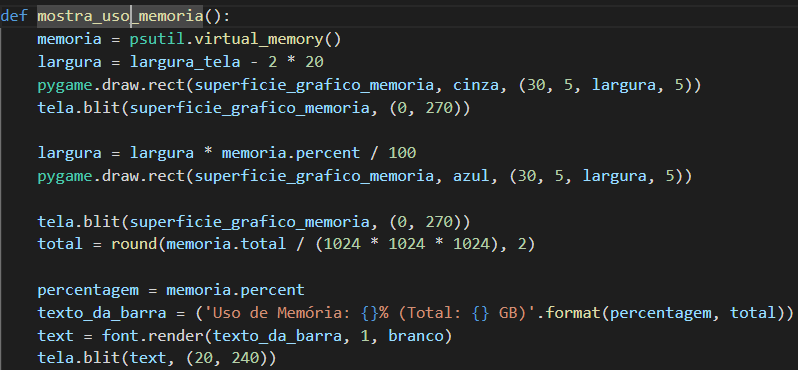
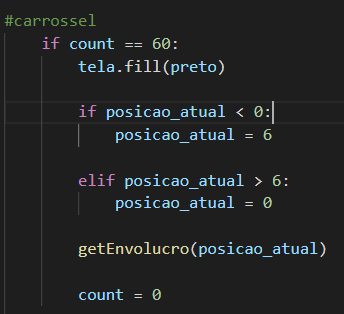


Figura Código para apresentar o consumo de memória

### Correção carrossel

Foi identificado um BUG presente na incrementação da variável global “**posicao\_atual**” onde não era verificado se essa variável era igual ao maior ou menor número de telas existentes. Sendo assim o código corrigido ficou da seguinte maneira:



# Capítulo 4

Esta sessão contempla todas as bibliotecas e funções utilizadas para atender aos requisitos da versão 4 de projeto de bloco.

## Objetivo

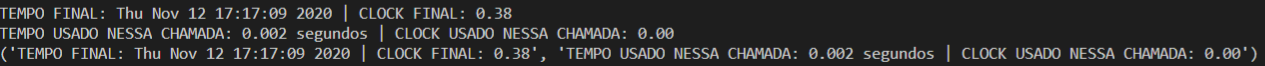
* Utilizar o módulo ‘sched’ para chamar as funções criadas no TP4 que retornam as informações sobre diretórios e arquivos.
* Realizar um escalonamento das chamadas das funções com o módulo ‘sched’ e medir o tempo total utilizado por cada chamada com o módulo ‘time’. Você pode escolher com quais funções do seu projeto realizar o escalonamento, deixando indicado no relatório.

## Bibliotecas utilizadas

* Pygame
* Psutil
* Cpuinfo
* Platform
* Subprocess
* Os
* Time
* Socket
* Nmap

## Desenvolvimento

Para atender ao requisito da utilização do módulo “***shed***”, foi criado uma função chamada “***get\_shed\_sheduler\_arquivos()***”, que tem por finalidade medir o tempo de execução de uma função. Após instanciar a configuração do “***scheduler***”, faz se necessário utilizar a função “***enter()***”, a mesma recebe como parâmetro a função que deverá ser monitorada. Entretanto para se obter o resultado corretamente, é necessário criar variáveis auxiliares para medir o tempo passado. O resultado exibido no console, mediante as medições foi:

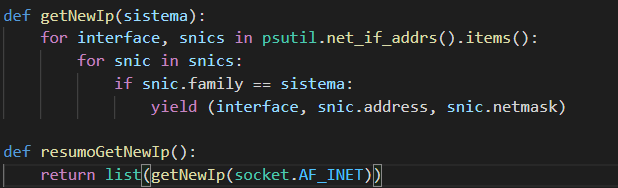


## Problemas observados:

* No capitulo passado, foi solicitado a listagem dos processos em execução e não iniciar um processo como foi feito.

## Soluções de problemas anteriores:

* Correção obtenção IPs da máquina do usuário:



Resultado obtido:

# Capítulo 5

Esta sessão contempla todas as bibliotecas e funções utilizadas para atender aos requisitos da versão 5 de projeto de bloco.

## Objetivo

* Criar funções que retornem ou apresentem informações sobre as máquinas pertencentes à sub rede do IP específico
* Criar funções que retornem os status das portas dos diferentes IPs obtidos nessa sub rede.

## Bibliotecas utilizadas

* Pygame
* Psutil
* Cpuinfo
* Platform
* Subprocess
* Os
* Time
* Socket
* Sched
* Nmap
* Threading
* Time

## Desenvolvimento

### Infraestrutura

Para obter sucesso durante a varredura dos hosts da sub rede foi necessário ter instalado previamente o software Nmap além da biblioteca nmap.

Para obter sucesso durante a varredura dos hosts pertencentes a rede foi necessária ter previamente o Ip da máquina do usuário. Em seguida, realizar um ping em todos os ips possíveis. Os hosts que responderam ao procedimento são postos em uma lista e em seguida é feito a verificação das portas ativas do mesmo. Para realizar essa etapa, foi utilizado a biblioteca ***nmap.PortScanner().*** Em seguida é utilizado a função **scan** para obter o status da porta.

## Problemas observados

* Ao executar a aplicação, o software entregue apresenta uma tela preta. Essa tela representa o travamento da aplicação ao iniciar a varredura dos hosts da rede. Uma possível solução para essa questão, seria iniciar a leitura da rede em um thread liberando o processo e informar ao usuário que os dados da rede estão sendo coletados.

## Soluções de problemas anteriores

* Processos em execução

Para solucionar esse problema, foram criadas duas classes: Processo e Porta.

A classe Processo possui os seguintes atributos:

* + Pid
  + Nome
  + Percentual\_uso
  + Memoria\_usada
  + Threads\_processo
  + Tempo\_usuario
  + Data\_criacao

Já a classe Porta, possui os seguintes atributos:

* + Porta
  + State

Dessa forma, para se obter os dados dos processos em execução, foi utilizado a biblioteca **psutil** e a função **pids()**. Os PIDs obtidos foram postos em uma lista. Por fim, foi utilizado a biblioteca **psutil** novamente juntamente com a função ***Process*** passando como paramento o PID obtido. Os dados propostos foram obtidos através das seguintes funções:

* + Memory\_percent()
  + Memory\_info().rss
  + Num\_threads()
  + Cpu\_time().user
  + Create\_time

# Capítulo 6

Esta sessão contempla todas as bibliotecas e funções utilizadas para atender aos requisitos da versão 6 de projeto de bloco.

## Objetivo

* Crie uma ou mais funções que retornem ou apresentem as seguintes informações de redes: IP, *gateway*, máscara de sub rede.
* Crie uma ou mais funções que retornem ou apresentem as seguintes informações de redes: Uso de dados de rede por interface.
* Crie uma ou mais funções que retornem ou apresentem as seguintes informações de redes: Uso de dados de rede por processos.

## Bibliotecas

* Pygame
* Psutil
* Cpuinfo
* Platform
* Subprocess
* Os
* Time
* Socket
* Sched
* Nmap
* Threading
* Time

## Desenvolvimento

Para atender aos requisitos desse capitulo, foram criadas as seguintes classes:

* Host
* Porta
* Trafego

Foi criado também uma classe chamada **ThreadRede,** que será responsável por obter os dados para os objetos citados sem que haja o travamento da aplicação.

A obtenção das máquinas da rede é feita utilizando a biblioteca **nmap** através da função **get\_hosts\_rede()**, que realiza a função **PING** na função **retorna\_codigo\_ping(),** em todos os ips possíveis da rede.

Com os hosts ativos na rede em mãos, é utilizado a função **detalhar\_host()** que é responsável por utilizar a função **nmap.PortScanner()** para obter a situação das portas do mesmo.

Entretanto, para se obter o trafego dos ips ativos da máquina do usuário, foi utilizado através da função **psutil.net\_io\_counters(),** em seguida os dados foram organizados nas suas classes respectivas.

## Problemas observados:

## Soluções de problemas anteriores:

* Travamento da aplicação ao inicializar:

Para solucionar esse problema, foi criado uma classe chamada **ThreadRede**. Ela herda da biblioteca **threading.Thread** nela foi criado uma função chamado **run**, que é responsável por inicializar a leitura dos hosts da rede, através da função **get\_hosts()**. Foi abordado essa solução, pois executando essa varredura em um thread separado, o software será liberado permitindo então a execução de outras funções.

# Capítulo 7

Esta sessão contempla todas as bibliotecas e funções utilizadas para atender aos requisitos da versão 7 de projeto de bloco.

## Objetivo

* Implementar ao menos 2 tipos de obtenção de informação de um computador, conforme feito nos TPs anteriores, mas agora no servidor. Você pode implementar todas que foram requisitadas nos TPs anteriores, mas não é necessário no momento.
* Implemente um programa cliente que requisite tais informações ao programa servidor e exiba os dados usando texto formatado ou de forma visual (com PyGame, por exemplo).
* Implemente um programa servidor que receba tais requisições do cliente, as obtenha na própria máquina e as envie ao cliente.

## Bibliotecas utilizadas

### Cliente

* Pygame
* Datetime
* Socket
* Pickle
* Os
* Threading

### Servidor

* Socket
* Psutil
* Pickle
* Cpuinfo
* Threading
* Time
* Sched
* Os
* Platform
* Subprocess
* Nmap
* Math

## Desenvolvimento

Para atender aos requisitos dessa versão, foram criadas duas aplicações:

* Servidor
  + Responsável por obter as informações da máquina e enviar da máquina em execução.

Manipula as classes:

* Host
* Porta
* Processo
* Arquivo
* CPU
* Memoria
* Disco
* Trafego
* Resumo

Manipula as Threads

* ThreadIps
* ThreadRede
* ThreadDisco
* ThreadCpu
* ThreadSched
* ThreadTrafegoRede
* ThreadMemoria

As Threads são iniciadas e nunca mais param. Elas são responsáveis por manter a variável global **variáveis,** que possui as seguintes chaves:

* Cpu
* Memoria
* Disco
* Processo
* Arquivos
* Sched
* Ips
* Hots\_detalhado
* Trafego

A obtenção dos dados é feita através da biblioteca **socket** que fica escutando a porta **9999**.

O servidor response as seguintes chamadas:

* Fim
* Disco
* Cpu
* Arquivos
* Ips
* Rede
* Trafego
* Memoria
* Processo
* Resumo

A biblioteca **pickle** é responsável por serializar os dados e enviar como resposta.

* + Cliente
    - Responsável por solicitar as informações para a máquina servidor e exibir pro usuário.
* Cliente
  + Responsável por solicitar as informações ao servidor e exibir pro usuário.
  + Responsável por gravar em um arquivo de log os dados, horário da solicitação, horário da resposta

Manipula os seguintes threads:

* ThreadLog

## Problemas observados

## Soluções de problemas anteriores

# Capítulo 8

## Quais as diferenças de arquitetura de CPU

## O que é processador?

## Quais as diferenças entre núcleos físicos e núcleos lógicos. Em que influência no processamento a utilização de núcleos lógicos

## Quais as diferenças dos módulos ‘OS’ E ‘PSUTIL’

## Quais as diferenças entre ‘clock da CPU’ e ‘Tempo Real’

# Anexos

## Código fonte v1

import pygame

import psutil

import cpuinfo

import platform

import os

import time

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Informações de CPU")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

#########################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(cinza)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width()- 2\*y - (num\_cpu+1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, vermelho, (d, y, larg, alt))

        pygame.draw.rect(s, azul, (d, y, larg, (1-i/100)\*alt))

        d = d + larg + desl

        tela.blit(s, (0, 140))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

########################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + "%"

    total\_disco = str(round(disco.used/(1024\*\*3), 2)) + "GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco():

    disco = psutil.disk\_usage('/')

    larg = largura\_tela - 2\*20

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 50, larg/2, 70))

    larg = larg \* disco.percent/100

    pygame.draw.rect(superficie\_grafico\_disco, Dim, (20, 50, larg/2, 70))

    total = round(disco.total/(1024\*\*3), 2)

    texto\_barra = "Uso de Disco: (Total: " + str(total) + "GB):"

    texto = font.render(texto\_barra, 1, branco)

    superficie\_grafico\_disco.blit(texto, (20, 10))

    tela.blit(superficie\_grafico\_disco, (0, 200))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco()

#########################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    mem = psutil.virtual\_memory()

    larg = largura\_tela - 2\*20

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (20, 50, larg/2, 70))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    larg = larg \* mem.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, preto, (20, 5, larg, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(mem.total / (1024 \* 1024 \* 1024), 2)

    percent = mem.percent

    msg = ('Uso de Memória: {}% (Total: {} GB)'.format(percent, total))

    text = font.render(msg, 1, branco)

    tela.blit(text, (20, 270))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

#########################################################################

def getIp():

    plataforma = platform.system()

    dic\_interfaces = psutil.net\_if\_addrs()

    if plataforma == 'Linux':

        ip = (dic\_interfaces['wlp3s0'][0].address)

        return ip

    elif plataforma == "Windows":

        ip = (dic\_interfaces['Ethernet'][1].address)

        return ip

    else:

        ip = (dic\_interfaces['wlp3s0'][0].address)

        return ip

def mostrar\_texto\_rede():

  plataforma = "IP: " + getIp()

  mostra\_texto(superficie\_info\_rede, plataforma, 10)

  tela.blit(superficie\_info\_rede, (20, 0))

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

########################################################################

def getEnvolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

#########################################################################

posicao\_atual = 0

posicao\_cpu = 0

posicao\_mem = 1

posicao\_dis = 2

posicao\_red = 3

posicao\_res = 3

#########################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

#carrossel

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < posicao\_cpu:

            posicao\_atual = 3

        elif posicao\_atual > posicao\_res:

            posicao\_atual = 0

        getEnvolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte v2

import pygame

import psutil

import cpuinfo

import platform

import os

import time

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Informações de CPU")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

#########################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(cinza)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width()- 2\*y - (num\_cpu+1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, vermelho, (d, y, larg, alt))

        pygame.draw.rect(s, azul, (d, y, larg, (1-i/100)\*alt))

        d = d + larg + desl

        tela.blit(s, (0, 140))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

#########################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + "%"

    total\_disco = str(round(disco.used/(1024\*\*3), 2)) + "GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco():

    disco = psutil.disk\_usage('/')

    larg = largura\_tela - 2\*20

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 50, larg/2, 70))

    larg = larg \* disco.percent/100

    pygame.draw.rect(superficie\_grafico\_disco, Dim, (20, 50, larg/2, 70))

    total = round(disco.total/(1024\*\*3), 2)

    texto\_barra = "Uso de Disco: (Total: " + str(total) + "GB):"

    texto = font.render(texto\_barra, 1, branco)

    superficie\_grafico\_disco.blit(texto, (20, 10))

    tela.blit(superficie\_grafico\_disco, (0, 200))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco()

#########################################################################

def resumo():

    superficie\_info\_resumo.fill(branco)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "disco\_livre", "Disco livre:", 10)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "memoria\_livre", "Memoria livre:", 30)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "ip\_rede", "IP:", 50)

    tela.blit(superficie\_info\_resumo, (0, 0))

def mostrar\_texto\_resumo(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    memoria\_livre = str(round(mem.available/(1024\*\*3), 2))

    ip = getIp()

    if chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

    elif chave == "memoria\_livre":

        s1.blit(font.render(memoria\_livre, True, preto), (155, pos\_y))

    elif chave == "ip\_rede":

        s1.blit(font.render(ip, True, preto), (155, pos\_y))

#########################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    mem = psutil.virtual\_memory()

    larg = largura\_tela - 2\*20

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (20, 50, larg/2, 70))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    larg = larg \* mem.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, preto, (20, 5, larg, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(mem.total / (1024 \* 1024 \* 1024), 2)

    percent = mem.percent

    msg = ('Uso de Memória: {}% (Total: {} GB)'.format(percent, total))

    text = font.render(msg, 1, branco)

    tela.blit(text, (20, 270))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

#########################################################################

def getIp():

    plataforma = platform.system()

    dic\_interfaces = psutil.net\_if\_addrs()

    if plataforma == 'Linux':

        ip = (dic\_interfaces['wlp3s0'][0].address)

        return ip

    elif plataforma == "Windows":

        ip = (dic\_interfaces['Ethernet'][1].address)

        return ip

    else:

        ip = (dic\_interfaces['wlp3s0'][0].address)

        return ip

def mostrar\_texto\_rede():

  plataforma = "IP: " + getIp()

  mostra\_texto(superficie\_info\_rede, plataforma, 10)

  tela.blit(superficie\_info\_rede, (20, 0))

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

#########################################################################

def getEnvolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

    elif posicao == 4:

        resumo()

#########################################################################

posicao\_atual = 0

posicao\_cpu = 0

posicao\_mem = 1

posicao\_dis = 2

posicao\_red = 3

posicao\_res = 4

#########################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

            elif event.key == pygame.K\_SPACE:

                posicao\_atual = 4

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < posicao\_cpu:

            posicao\_atual = 5

        elif posicao\_atual > posicao\_res:

            posicao\_atual = 0

        getEnvolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte v3

import pygame

import psutil

import cpuinfo

import subprocess

import os

import time

import socket

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Projeto de bloco - Carlos Henrique")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_ips = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

#########################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(preto)

    capacidade = psutil.cpu\_percent(interval=1)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width() - 2 \* y - (num\_cpu + 1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, azul, (d, y + 20, larg, alt))

        pygame.draw.rect(s, Dim, (d, y + 20, larg, (1 - i / 100) \* alt))

        d = d + larg + desl

    text = font.render("Uso de CPU por núcleo total: " + str(capacidade) + "%", 1, branco)

    s.blit(text, (20, 5))

    tela.blit(s, (0, 200))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

#########################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('.')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + " %"

    total\_disco = str(round(disco.total / (1024\*\*3), 2)) + " GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + " GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco(eixo\_x, eixo\_y):

    disco = psutil.disk\_usage('.')

    total = round(disco.total / (1024 \* 1024 \* 1024), 2)

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_disco, cinza, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_x))

    largura = largura \* disco.percent / 100

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_y))

    percentagem = disco.percent

    texto\_da\_barra = ('Uso de Disco: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, eixo\_y))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco(410, 390)

#########################################################################

def resumo():

    superficie\_info\_resumo.fill(branco)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "disco\_livre", "Disco livre:", 10)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "memoria\_livre", "Memoria livre:", 30)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "ip\_rede", "IP:", 50)

    tela.blit(superficie\_info\_resumo, (0, 0))

def mostrar\_texto\_resumo(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    memoria\_livre = str(round(mem.available/(1024\*\*3), 2))

    ip = resumoGetNewIp()

    if chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

    elif chave == "memoria\_livre":

        s1.blit(font.render(memoria\_livre, True, preto), (155, pos\_y))

    elif chave == "ip\_rede":

        s1.blit(font.render(ip[0][1], True, preto), (155, pos\_y))

#########################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    memoria = psutil.virtual\_memory()

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_memoria, cinza, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    largura = largura \* memoria.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(memoria.total / (1024 \* 1024 \* 1024), 2)

    percentagem = memoria.percent

    texto\_da\_barra = ('Uso de Memória: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, 240))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

#########################################################################

def envolucro\_arquivos():

    arquivos = mostrar\_dados\_diretorio()

    apresenta\_dados(arquivos)

def mostrar\_dados\_diretorio():

    lista = os.listdir()

    dados\_organizados = {}

    for i in lista:

        dados\_organizados[i] = []

        dados\_organizados[i].append(os.stat(i).st\_size)

        dados\_organizados[i].append(os.stat(i).st\_ctime)

        dados\_organizados[i].append(os.stat(i).st\_mtime)

    tituloInfo = 'Arquivos do diretório:'

    titulo\_tamanho = '{:>5}'.format('Tamanho')

    titulo\_data\_criacao = '{:>30}'.format('Criação')

    titulo\_data\_modificacao = '{:>38}'.format('Modificação')

    titulo\_nome = '{:>38}'.format('Nome')

    titulo = titulo\_tamanho + titulo\_data\_criacao + titulo\_data\_modificacao + titulo\_nome

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (20, 20))

    textoTitulo = font.render(titulo, 1, branco)

    tela.blit(textoTitulo, (20, 60))

    return dados\_organizados

def apresenta\_dados(arquivos):

    espacos = 100

    for i in arquivos:

        tamanho\_arquivo = arquivos[i][0]/1024

        tamanho\_formatado = '{:>10}'.format(str('{:.2f}'.format(tamanho\_arquivo) + 'KB'))

        data\_criacao = '{:>30}'.format(time.ctime(arquivos[i][0]))

        time\_mod = '{:>30}'.format(time.ctime(arquivos[i][1]))

        nomeArquivo = '{:>30}'.format(i)

        textoArqDir = font.render(tamanho\_formatado + data\_criacao + time\_mod + nomeArquivo, 1, branco)

        tela.blit(textoArqDir, (15, espacos))

        espacos += 25

#########################################################################

def mostrar\_texto\_rede():

    espacos = 100

    for ip in ips:

        texto = font.render(ip[0] + ': ' + ip[1] + ' - ' + ip[2], 1, branco)

        tela.blit(texto, (15, espacos))

        espacos += 25

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

def getNewIp(sistema):

    for interface, snics in psutil.net\_if\_addrs().items():

        for snic in snics:

            if snic.family == sistema:

                yield (interface, snic.address, snic.netmask)

def resumoGetNewIp():

    return list(getNewIp(socket.AF\_INET))

meu\_ip = ''

hosts = []

ips = resumoGetNewIp()

#########################################################################

def info\_processos():

    pid = subprocess.Popen('cmd.exe').pid

    p = psutil.Process(pid)

    perc\_mem = '{:.2f}'.format(p.memory\_percent())

    mem = '{:.2f}'.format(p.memory\_info().rss/1024/1024)

    tituloInfo = 'Informações sobre processos:'

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (100, 160))

    texto1 = 'Nome: ' + p.name()

    texto01 = font.render(texto1, 1, branco)

    tela.blit(texto01, (100,200))

    texto2 = 'Executável: ' + p.exe()

    texto02 = font.render(texto2, 1, branco)

    tela.blit(texto02, (100,220))

    texto3 = 'Tempo de criação: ' + time.ctime(p.create\_time())

    texto03 = font.render(texto3, 1, branco)

    tela.blit(texto03, (100,240))

    texto4 = 'Tempo de usuário: ' + str(p.cpu\_times().user) + 's'

    texto04 = font.render(texto4, 1, branco)

    tela.blit(texto04, (100,260))

    texto5 = 'Tempo de sistema: ' + str(p.cpu\_times().system) + 's'

    texto05 = font.render(texto5, 1, branco)

    tela.blit(texto05, (100,280))

    texto6 = 'Percentual de uso de CPU: ' + str(p.cpu\_percent(interval=1.0)) + '%'

    texto06 = font.render(texto6, 1, branco)

    tela.blit(texto06, (100,300))

    texto7 = 'Percentual de uso de memória: ' + perc\_mem + '%'

    texto07 = font.render(texto7, 1, branco)

    tela.blit(texto07, (100,320))

    texto8 = 'Uso de memória: ' + mem + 'MB'

    texto08 = font.render(texto8, 1, branco)

    tela.blit(texto08, (100,340))

    texto9 = 'Número de threads: ' + str(p.num\_threads())

    texto09 = font.render(texto9, 1, branco)

    tela.blit(texto09, (100,360))

#########################################################################

def getEnvolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

    elif posicao == 4:

        envolucro\_arquivos()

    elif posicao == 5:

        info\_processos()

    elif posicao == 6:

        resumo()

#########################################################################

posicao\_atual = 0

#########################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

            elif event.key == pygame.K\_SPACE:

                posicao\_atual = 4

#carrossel

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < 0:

            posicao\_atual = 6

        elif posicao\_atual > 6:

            posicao\_atual = 0

        getEnvolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte v4

import pygame

import psutil

import cpuinfo

import platform

import subprocess

import os

import time

import socket

import nmap

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Projeto de bloco - Carlos Henrique")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

class Host:

    def \_\_init\_\_(self, ip, name):

        self.ip = ip

        self.name = name

        self.ports = []

class Port:

    def \_\_init\_\_(self, port, state):

        self.port = port

        self.state = state

meu\_ip = ''

hosts = []

#########################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(preto)

    capacidade = psutil.cpu\_percent(interval=1)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width() - 2 \* y - (num\_cpu + 1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, azul, (d, y + 20, larg, alt))

        pygame.draw.rect(s, Dim, (d, y + 20, larg, (1 - i / 100) \* alt))

        d = d + larg + desl

    text = font.render("Uso de CPU por núcleo total: " + str(capacidade) + "%", 1, branco)

    s.blit(text, (20, 5))

    tela.blit(s, (0, 200))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

#########################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('.')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + " %"

    total\_disco = str(round(disco.total / (1024\*\*3), 2)) + " GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + " GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco(eixo\_x, eixo\_y):

    disco = psutil.disk\_usage('.')

    total = round(disco.total / (1024 \* 1024 \* 1024), 2)

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_disco, cinza, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_x))

    largura = largura \* disco.percent / 100

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_y))

    percentagem = disco.percent

    texto\_da\_barra = ('Uso de Disco: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, eixo\_y))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco(410, 390)

#########################################################################

def resumo():

    superficie\_info\_resumo.fill(branco)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "disco\_livre", "Disco livre:", 10)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "memoria\_livre", "Memoria livre:", 30)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "ip\_rede", "IP:", 50)

    tela.blit(superficie\_info\_resumo, (0, 0))

def mostrar\_texto\_resumo(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    memoria\_livre = str(round(mem.available/(1024\*\*3), 2))

    ip = resumoGetNewIp()

    if chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

    elif chave == "memoria\_livre":

        s1.blit(font.render(memoria\_livre, True, preto), (155, pos\_y))

    elif chave == "ip\_rede":

        s1.blit(font.render(ip[0][1], True, preto), (155, pos\_y))

#########################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    memoria = psutil.virtual\_memory()

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_memoria, cinza, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    largura = largura \* memoria.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(memoria.total / (1024 \* 1024 \* 1024), 2)

    percentagem = memoria.percent

    texto\_da\_barra = ('Uso de Memória: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, 240))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

#########################################################################

def envolucro\_arquivos():

    arquivos = mostrar\_dados\_diretorio()

    apresenta\_dados(arquivos)

def mostrar\_dados\_diretorio():

    lista = os.listdir()

    dados\_organizados = {}

    for i in lista:

        dados\_organizados[i] = []

        dados\_organizados[i].append(os.stat(i).st\_size)

        dados\_organizados[i].append(os.stat(i).st\_ctime)

        dados\_organizados[i].append(os.stat(i).st\_mtime)

    tituloInfo = 'Arquivos do diretório:'

    titulo\_tamanho = '{:>5}'.format('Tamanho')

    titulo\_data\_criacao = '{:>30}'.format('Criação')

    titulo\_data\_modificacao = '{:>38}'.format('Modificação')

    titulo\_nome = '{:>38}'.format('Nome')

    titulo = titulo\_tamanho + titulo\_data\_criacao + titulo\_data\_modificacao + titulo\_nome

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (20, 20))

    textoTitulo = font.render(titulo, 1, branco)

    tela.blit(textoTitulo, (20, 60))

    return dados\_organizados

def apresenta\_dados(arquivos):

    espacos = 100

    for i in arquivos:

        tamanho\_arquivo = arquivos[i][0]/1024

        tamanho\_formatado = '{:>10}'.format(str('{:.2f}'.format(tamanho\_arquivo) + 'KB'))

        data\_criacao = '{:>30}'.format(time.ctime(arquivos[i][0]))

        time\_mod = '{:>30}'.format(time.ctime(arquivos[i][1]))

        nomeArquivo = '{:>30}'.format(i)

        textoArqDir = font.render(tamanho\_formatado + data\_criacao + time\_mod + nomeArquivo, 1, branco)

        tela.blit(textoArqDir, (15, espacos))

        espacos += 25

#########################################################################

def mostrar\_texto\_rede():

    espacos = 100

    for ip in ips:

        texto = font.render(ip[0] + ': ' + ip[1] + ' - ' + ip[2], 1, branco)

        tela.blit(texto, (15, espacos))

        espacos += 25

    titulo = font.render("\*\* Rede \*\*" ,1, azul)

    tela.blit(titulo,(15, 20))

    titulo2 = font.render("\*\* Hosts da Rede \*\*" ,1, azul)

    tela.blit(titulo2,(15, 190))

    for host in hosts:

        host\_name = ""

        if host.name != "":

            host\_name = host.name

        else:

            host\_name = "NÃO LOCALIZADO"

        cor = ""

        if host\_name != "NÃO LOCALIZADO":

            cor = (255, 255, 255)

        else:

            cor = (255, 0, 0)

        texto = font.render(host.ip + ': Nome: ' + host\_name, 1, cor)

        tela.blit(texto, (15, espacos + 5))

        espacos += 15

        for porta in host.ports:

            detalhe\_porta = font.render("Porta: " + str(porta.port) + " - Estado: " + porta.state ,1, branco)

            tela.blit(detalhe\_porta, (15, espacos))

            espacos += 15

        espacos += 10

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

def getNewIp(sistema):

    for interface, snics in psutil.net\_if\_addrs().items():

        for snic in snics:

            if snic.family == sistema:

                yield (interface, snic.address, snic.netmask)

def resumoGetNewIp():

    return list(getNewIp(socket.AF\_INET))

def retorna\_codigo\_ping(hostname):

    """Usa o utilitario ping do sistema operacional para encontrar   o host. ('-c 5') indica, em sistemas linux, que deve mandar 5   pacotes. ('-W 3') indica, em sistemas linux, que deve esperar 3   milisegundos por uma resposta. Esta funcao retorna o codigo de   resposta do ping """

    plataforma = platform.system()

    args = []

    if plataforma == "Windows":

        args = ["ping", "-n", "1", "-l", "1", "-w", "100", hostname]

    else:

        args = ['ping', '-c', '1', '-W', '1', hostname]

    ret\_cod = subprocess.call(args, stdout=open(os.devnull, 'w'), stderr=open(os.devnull, 'w'))

    return ret\_cod

def verifica\_hosts\_validos(base\_ip):

    """Verifica todos os host com a base\_ip entre 1 e 255 retorna uma lista com todos os host que tiveram resposta 0 (ativo)"""

    print("Mapeando\r")

    host\_validos = []

    return\_codes = dict()

    for i in range(1, 255):

        return\_codes[base\_ip + '{0}'.format(i)] = retorna\_codigo\_ping(base\_ip + '{0}'.format(i))

        if i %20 ==0:

            print(".", end = "")

        if return\_codes[base\_ip + '{0}'.format(i)] == 0:

            host\_validos.append(base\_ip + '{0}'.format(i))

    return host\_validos

def detalhar\_host(host\_validos):

    """Obtendo nome do host"""

    nm = nmap.PortScanner()

    for host in host\_validos:

        print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r')

        try:

            nm.scan(host)

            host\_ = Host(host, nm[host].hostname())

            #print('O IP', host, 'possui o nome', nm[host].hostname())

            for proto in nm[host].all\_protocols():

                print('----------')

                print('Protocolo : %s' % proto)

            lport = nm[host][proto].keys()

            for port in lport:

                port\_ = Port(port, nm[host][proto][port]['state'])

                host\_.ports.append(port\_)

                #print ('Porta: %s\t Estado: %s' % (port, nm[host][proto][port]['state']))

        except:

            print(':rocket: Exception')

            pass

        hosts.append(host\_)

executando = False

executou = False

ips = resumoGetNewIp()

def envolucro\_detalhar\_host():

    executando = True

    print('Iniciando coleta de dados da rede:', executando)

    meu\_ip = ips[0][1]

    print('Ip que será utilizado como base', meu\_ip)

    ip\_string = meu\_ip

    ip\_lista = ip\_string.split('.')

    base\_ip = ".".join(ip\_lista[0:3]) + '.'

    print("O teste será feito na sub rede: ", base\_ip)

    hosts\_localizados = verifica\_hosts\_validos(base\_ip)

    print ("Os host válidos são: ", hosts\_localizados)

    print('Verifica nome do host\r')

    detalhar\_host(hosts\_localizados)

    executou = True

    print('Processo finalizado', executou)

#########################################################################

def info\_processos():

    pid = subprocess.Popen('cmd.exe').pid

    p = psutil.Process(pid)

    perc\_mem = '{:.2f}'.format(p.memory\_percent())

    mem = '{:.2f}'.format(p.memory\_info().rss/1024/1024)

    tituloInfo = 'Informações sobre processos:'

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (100, 160))

    texto1 = 'Nome: ' + p.name()

    texto01 = font.render(texto1, 1, branco)

    tela.blit(texto01, (100,200))

    texto2 = 'Executável: ' + p.exe()

    texto02 = font.render(texto2, 1, branco)

    tela.blit(texto02, (100,220))

    texto3 = 'Tempo de criação: ' + time.ctime(p.create\_time())

    texto03 = font.render(texto3, 1, branco)

    tela.blit(texto03, (100,240))

    texto4 = 'Tempo de usuário: ' + str(p.cpu\_times().user) + 's'

    texto04 = font.render(texto4, 1, branco)

    tela.blit(texto04, (100,260))

    texto5 = 'Tempo de sistema: ' + str(p.cpu\_times().system) + 's'

    texto05 = font.render(texto5, 1, branco)

    tela.blit(texto05, (100,280))

    texto6 = 'Percentual de uso de CPU: ' + str(p.cpu\_percent(interval=1.0)) + '%'

    texto06 = font.render(texto6, 1, branco)

    tela.blit(texto06, (100,300))

    texto7 = 'Percentual de uso de memória: ' + perc\_mem + '%'

    texto07 = font.render(texto7, 1, branco)

    tela.blit(texto07, (100,320))

    texto8 = 'Uso de memória: ' + mem + 'MB'

    texto08 = font.render(texto8, 1, branco)

    tela.blit(texto08, (100,340))

    texto9 = 'Número de threads: ' + str(p.num\_threads())

    texto09 = font.render(texto9, 1, branco)

    tela.blit(texto09, (100,360))

#########################################################################

def getEnvolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

        if len(hosts) == 0 and not executando and not executou:

            envolucro\_detalhar\_host()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

    elif posicao == 4:

        envolucro\_arquivos()

    elif posicao == 5:

        info\_processos()

    elif posicao == 6:

        resumo()

#########################################################################

posicao\_atual = 0

#########################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

            elif event.key == pygame.K\_SPACE:

                posicao\_atual = 6

#carrossel

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < 0:

            posicao\_atual = 6

        elif posicao\_atual > 6:

            posicao\_atual = 0

        getEnvolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte v5

import pygame

import psutil

import cpuinfo

import platform

import subprocess

import os

import time

import socket

import nmap

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Projeto de bloco - Carlos Henrique")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

#superficie\_info\_ips = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

class Host:

    def \_\_init\_\_(self, ip, name):

        self.ip = ip

        self.name = name

        self.ports = []

class Port:

    def \_\_init\_\_(self, port, state):

        self.port = port

        self.state = state

meu\_ip = ''

hosts = []

#########################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(preto)

    capacidade = psutil.cpu\_percent(interval=1)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width() - 2 \* y - (num\_cpu + 1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, azul, (d, y + 20, larg, alt))

        pygame.draw.rect(s, Dim, (d, y + 20, larg, (1 - i / 100) \* alt))

        d = d + larg + desl

    text = font.render("Uso de CPU por núcleo total: " + str(capacidade) + "%", 1, branco)

    s.blit(text, (20, 5))

    tela.blit(s, (0, 200))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

#########################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('.')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + " %"

    total\_disco = str(round(disco.total / (1024\*\*3), 2)) + " GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + " GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco(eixo\_x, eixo\_y):

    disco = psutil.disk\_usage('.')

    total = round(disco.total / (1024 \* 1024 \* 1024), 2)

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_disco, cinza, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_x))

    largura = largura \* disco.percent / 100

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_y))

    percentagem = disco.percent

    texto\_da\_barra = ('Uso de Disco: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, eixo\_y))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco(410, 390)

#########################################################################

def resumo():

    superficie\_info\_resumo.fill(branco)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "disco\_livre", "Disco livre:", 10)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "memoria\_livre", "Memoria livre:", 30)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "ip\_rede", "IP:", 50)

    tela.blit(superficie\_info\_resumo, (0, 0))

def mostrar\_texto\_resumo(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    memoria\_livre = str(round(mem.available/(1024\*\*3), 2))

    ip = resumoGetNewIp()

    if chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

    elif chave == "memoria\_livre":

        s1.blit(font.render(memoria\_livre, True, preto), (155, pos\_y))

    elif chave == "ip\_rede":

        s1.blit(font.render(ip[0][1], True, preto), (155, pos\_y))

#########################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    memoria = psutil.virtual\_memory()

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_memoria, cinza, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    largura = largura \* memoria.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(memoria.total / (1024 \* 1024 \* 1024), 2)

    percentagem = memoria.percent

    texto\_da\_barra = ('Uso de Memória: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, 240))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

#########################################################################

def envolucro\_arquivos():

    arquivos = mostrar\_dados\_diretorio()

    apresenta\_dados(arquivos)

def mostrar\_dados\_diretorio():

    lista = os.listdir()

    dados\_organizados = {}

    for i in lista:

        dados\_organizados[i] = []

        dados\_organizados[i].append(os.stat(i).st\_size)

        dados\_organizados[i].append(os.stat(i).st\_ctime)

        dados\_organizados[i].append(os.stat(i).st\_mtime)

    tituloInfo = 'Arquivos do diretório:'

    titulo\_tamanho = '{:>5}'.format('Tamanho')

    titulo\_data\_criacao = '{:>30}'.format('Criação')

    titulo\_data\_modificacao = '{:>38}'.format('Modificação')

    titulo\_nome = '{:>38}'.format('Nome')

    titulo = titulo\_tamanho + titulo\_data\_criacao + titulo\_data\_modificacao + titulo\_nome

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (20, 20))

    textoTitulo = font.render(titulo, 1, branco)

    tela.blit(textoTitulo, (20, 60))

    return dados\_organizados

def apresenta\_dados(arquivos):

    espacos = 100

    for i in arquivos:

        tamanho\_arquivo = arquivos[i][0]/1024

        tamanho\_formatado = '{:>10}'.format(str('{:.2f}'.format(tamanho\_arquivo) + 'KB'))

        data\_criacao = '{:>30}'.format(time.ctime(arquivos[i][0]))

        time\_mod = '{:>30}'.format(time.ctime(arquivos[i][1]))

        nomeArquivo = '{:>30}'.format(i)

        textoArqDir = font.render(tamanho\_formatado + data\_criacao + time\_mod + nomeArquivo, 1, branco)

        tela.blit(textoArqDir, (15, espacos))

        espacos += 25

#########################################################################

def mostrar\_texto\_rede():

    espacos = 100

    for ip in ips:

        texto = font.render(ip[0] + ': ' + ip[1] + ' - ' + ip[2], 1, branco)

        tela.blit(texto, (15, espacos))

        espacos += 25

    titulo = font.render("\*\* Rede \*\*" ,1, azul)

    tela.blit(titulo,(15, 20))

    titulo2 = font.render("\*\* Hosts da Rede \*\*" ,1, azul)

    tela.blit(titulo2,(15, 190))

    for host in hosts:

        host\_name = ""

        if host.name != "":

            host\_name = host.name

        else:

            host\_name = "NÃO LOCALIZADO"

        cor = ""

        if host\_name != "NÃO LOCALIZADO":

            cor = (255, 255, 255)

        else:

            cor = (255, 0, 0)

        texto = font.render(host.ip + ': Nome: ' + host\_name, 1, cor)

        tela.blit(texto, (15, espacos + 5))

        espacos += 15

        for porta in host.ports:

            detalhe\_porta = font.render("Porta: " + str(porta.port) + " - Estado: " + porta.state ,1, branco)

            tela.blit(detalhe\_porta, (15, espacos))

            espacos += 15

        espacos += 10

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

def getNewIp(sistema):

    for interface, snics in psutil.net\_if\_addrs().items():

        for snic in snics:

            if snic.family == sistema:

                yield (interface, snic.address, snic.netmask)

def resumoGetNewIp():

    return list(getNewIp(socket.AF\_INET))

def retorna\_codigo\_ping(hostname):

    """Usa o utilitario ping do sistema operacional para encontrar   o host. ('-c 5') indica, em sistemas linux, que deve mandar 5   pacotes. ('-W 3') indica, em sistemas linux, que deve esperar 3   milisegundos por uma resposta. Esta funcao retorna o codigo de   resposta do ping """

    plataforma = platform.system()

    args = []

    if plataforma == "Windows":

        args = ["ping", "-n", "1", "-l", "1", "-w", "100", hostname]

    else:

        args = ['ping', '-c', '1', '-W', '1', hostname]

    ret\_cod = subprocess.call(args, stdout=open(os.devnull, 'w'), stderr=open(os.devnull, 'w'))

    return ret\_cod

def verifica\_hosts\_validos(base\_ip):

    """Verifica todos os host com a base\_ip entre 1 e 255 retorna uma lista com todos os host que tiveram resposta 0 (ativo)"""

    print("Mapeando\r")

    host\_validos = []

    return\_codes = dict()

    for i in range(1, 255):

        return\_codes[base\_ip + '{0}'.format(i)] = retorna\_codigo\_ping(base\_ip + '{0}'.format(i))

        if i %20 ==0:

            print(".", end = "")

        if return\_codes[base\_ip + '{0}'.format(i)] == 0:

            host\_validos.append(base\_ip + '{0}'.format(i))

    print("\nMapeamento pronto...")

    return host\_validos

def detalhar\_host(host\_validos):

    """Obtendo nome do host"""

    nm = nmap.PortScanner()

    for host in host\_validos:

        print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r')

        try:

            nm.scan(host)

            host\_ = Host(host, nm[host].hostname())

            #print('O IP', host, 'possui o nome', nm[host].hostname())

            for proto in nm[host].all\_protocols():

                print('----------')

                print('Protocolo : %s' % proto)

            lport = nm[host][proto].keys()

            for port in lport:

                port\_ = Port(port, nm[host][proto][port]['state'])

                host\_.ports.append(port\_)

                #print ('Porta: %s\t Estado: %s' % (port, nm[host][proto][port]['state']))

        except:

            print(':rocket: Exception')

            pass

        hosts.append(host\_)

executando = False

executou = False

ips = resumoGetNewIp()

def envolucro\_detalhar\_host():

    executando = True

    print('Iniciando coleta de dados da rede:', executando)

    meu\_ip = ips[0][1]

    print('Ip que será utilizado como base', meu\_ip)

    ip\_string = meu\_ip

    ip\_lista = ip\_string.split('.')

    base\_ip = ".".join(ip\_lista[0:3]) + '.'

    print("O teste será feito na sub rede: ", base\_ip)

    hosts\_localizados = verifica\_hosts\_validos(base\_ip)

    print ("Os host válidos são: ", hosts\_localizados)

    print('Verifica nome do host\r')

    detalhar\_host(hosts\_localizados)

    executou = True

    print('Processo finalizado', executou)

#########################################################################

def info\_processos():

    pid = subprocess.Popen('cmd.exe').pid

    p = psutil.Process(pid)

    perc\_mem = '{:.2f}'.format(p.memory\_percent())

    mem = '{:.2f}'.format(p.memory\_info().rss/1024/1024)

    tituloInfo = 'Informações sobre processos:'

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (100, 160))

    texto1 = 'Nome: ' + p.name()

    texto01 = font.render(texto1, 1, branco)

    tela.blit(texto01, (100,200))

    texto2 = 'Executável: ' + p.exe()

    texto02 = font.render(texto2, 1, branco)

    tela.blit(texto02, (100,220))

    texto3 = 'Tempo de criação: ' + time.ctime(p.create\_time())

    texto03 = font.render(texto3, 1, branco)

    tela.blit(texto03, (100,240))

    texto4 = 'Tempo de usuário: ' + str(p.cpu\_times().user) + 's'

    texto04 = font.render(texto4, 1, branco)

    tela.blit(texto04, (100,260))

    texto5 = 'Tempo de sistema: ' + str(p.cpu\_times().system) + 's'

    texto05 = font.render(texto5, 1, branco)

    tela.blit(texto05, (100,280))

    texto6 = 'Percentual de uso de CPU: ' + str(p.cpu\_percent(interval=1.0)) + '%'

    texto06 = font.render(texto6, 1, branco)

    tela.blit(texto06, (100,300))

    texto7 = 'Percentual de uso de memória: ' + perc\_mem + '%'

    texto07 = font.render(texto7, 1, branco)

    tela.blit(texto07, (100,320))

    texto8 = 'Uso de memória: ' + mem + 'MB'

    texto08 = font.render(texto8, 1, branco)

    tela.blit(texto08, (100,340))

    texto9 = 'Número de threads: ' + str(p.num\_threads())

    texto09 = font.render(texto9, 1, branco)

    tela.blit(texto09, (100,360))

#########################################################################

def get\_envolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

        if len(hosts) == 0 and not executando and not executou:

            envolucro\_detalhar\_host()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

    elif posicao == 4:

        envolucro\_arquivos()

    elif posicao == 5:

        info\_processos()

    elif posicao == 6:

        resumo()

#########################################################################

posicao\_atual = 0

#########################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

            elif event.key == pygame.K\_SPACE:

                posicao\_atual = 4

#carrossel

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < 0:

            posicao\_atual = 6

        elif posicao\_atual > 6:

            posicao\_atual = 0

        get\_envolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte v6

import pygame

import psutil

import cpuinfo

import platform

import subprocess

import os

import time

import socket

import nmap

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Projeto de bloco - Carlos Henrique")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

#superficie\_info\_ips = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

class Host:

    def \_\_init\_\_(self, ip, name):

        self.ip = ip

        self.name = name

        self.ports = []

class Port:

    def \_\_init\_\_(self, port, state):

        self.port = port

        self.state = state

meu\_ip = ''

hosts = []

#########################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(preto)

    capacidade = psutil.cpu\_percent(interval=1)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width() - 2 \* y - (num\_cpu + 1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, azul, (d, y + 20, larg, alt))

        pygame.draw.rect(s, Dim, (d, y + 20, larg, (1 - i / 100) \* alt))

        d = d + larg + desl

    text = font.render("Uso de CPU por núcleo total: " + str(capacidade) + "%", 1, branco)

    s.blit(text, (20, 5))

    tela.blit(s, (0, 200))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

#########################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('.')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + " %"

    total\_disco = str(round(disco.total / (1024\*\*3), 2)) + " GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + " GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco(eixo\_x, eixo\_y):

    disco = psutil.disk\_usage('.')

    total = round(disco.total / (1024 \* 1024 \* 1024), 2)

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_disco, cinza, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_x))

    largura = largura \* disco.percent / 100

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_y))

    percentagem = disco.percent

    texto\_da\_barra = ('Uso de Disco: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, eixo\_y))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco(410, 390)

#########################################################################

def resumo():

    superficie\_info\_resumo.fill(branco)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "disco\_livre", "Disco livre:", 10)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "memoria\_livre", "Memoria livre:", 30)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "ip\_rede", "IP:", 50)

    tela.blit(superficie\_info\_resumo, (0, 0))

def mostrar\_texto\_resumo(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    memoria\_livre = str(round(mem.available/(1024\*\*3), 2))

    ip = resumoGetNewIp()

    if chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

    elif chave == "memoria\_livre":

        s1.blit(font.render(memoria\_livre, True, preto), (155, pos\_y))

    elif chave == "ip\_rede":

        s1.blit(font.render(ip[0][1], True, preto), (155, pos\_y))

#########################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    memoria = psutil.virtual\_memory()

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_memoria, cinza, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    largura = largura \* memoria.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(memoria.total / (1024 \* 1024 \* 1024), 2)

    percentagem = memoria.percent

    texto\_da\_barra = ('Uso de Memória: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, 240))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

#########################################################################

def envolucro\_arquivos():

    arquivos = mostrar\_dados\_diretorio()

    apresenta\_dados(arquivos)

def mostrar\_dados\_diretorio():

    lista = os.listdir()

    dados\_organizados = {}

    for i in lista:

        dados\_organizados[i] = []

        dados\_organizados[i].append(os.stat(i).st\_size)

        dados\_organizados[i].append(os.stat(i).st\_ctime)

        dados\_organizados[i].append(os.stat(i).st\_mtime)

    tituloInfo = 'Arquivos do diretório:'

    titulo\_tamanho = '{:>5}'.format('Tamanho')

    titulo\_data\_criacao = '{:>30}'.format('Criação')

    titulo\_data\_modificacao = '{:>38}'.format('Modificação')

    titulo\_nome = '{:>38}'.format('Nome')

    titulo = titulo\_tamanho + titulo\_data\_criacao + titulo\_data\_modificacao + titulo\_nome

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (20, 20))

    textoTitulo = font.render(titulo, 1, branco)

    tela.blit(textoTitulo, (20, 60))

    return dados\_organizados

def apresenta\_dados(arquivos):

    espacos = 100

    for i in arquivos:

        tamanho\_arquivo = arquivos[i][0]/1024

        tamanho\_formatado = '{:>10}'.format(str('{:.2f}'.format(tamanho\_arquivo) + 'KB'))

        data\_criacao = '{:>30}'.format(time.ctime(arquivos[i][0]))

        time\_mod = '{:>30}'.format(time.ctime(arquivos[i][1]))

        nomeArquivo = '{:>30}'.format(i)

        textoArqDir = font.render(tamanho\_formatado + data\_criacao + time\_mod + nomeArquivo, 1, branco)

        tela.blit(textoArqDir, (15, espacos))

        espacos += 25

#########################################################################

def mostrar\_texto\_rede():

    espacos = 100

    for ip in ips:

        texto = font.render(ip[0] + ': ' + ip[1] + ' - ' + ip[2], 1, branco)

        tela.blit(texto, (15, espacos))

        espacos += 25

    titulo = font.render("\*\* Rede \*\*" ,1, azul)

    tela.blit(titulo,(15, 20))

    titulo2 = font.render("\*\* Hosts da Rede \*\*" ,1, azul)

    tela.blit(titulo2,(15, 190))

    for host in hosts:

        host\_name = ""

        if host.name != "":

            host\_name = host.name

        else:

            host\_name = "NÃO LOCALIZADO"

        cor = ""

        if host\_name != "NÃO LOCALIZADO":

            cor = (255, 255, 255)

        else:

            cor = (255, 0, 0)

        texto = font.render(host.ip + ': Nome: ' + host\_name, 1, cor)

        tela.blit(texto, (15, espacos + 5))

        espacos += 15

        for porta in host.ports:

            detalhe\_porta = font.render("Porta: " + str(porta.port) + " - Estado: " + porta.state ,1, branco)

            tela.blit(detalhe\_porta, (15, espacos))

            espacos += 15

        espacos += 10

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

def getNewIp(sistema):

    for interface, snics in psutil.net\_if\_addrs().items():

        for snic in snics:

            if snic.family == sistema:

                yield (interface, snic.address, snic.netmask)

def resumoGetNewIp():

    return list(getNewIp(socket.AF\_INET))

def retorna\_codigo\_ping(hostname):

    """Usa o utilitario ping do sistema operacional para encontrar   o host. ('-c 5') indica, em sistemas linux, que deve mandar 5   pacotes. ('-W 3') indica, em sistemas linux, que deve esperar 3   milisegundos por uma resposta. Esta funcao retorna o codigo de   resposta do ping """

    plataforma = platform.system()

    args = []

    if plataforma == "Windows":

        args = ["ping", "-n", "1", "-l", "1", "-w", "100", hostname]

    else:

        args = ['ping', '-c', '1', '-W', '1', hostname]

    ret\_cod = subprocess.call(args, stdout=open(os.devnull, 'w'), stderr=open(os.devnull, 'w'))

    return ret\_cod

def verifica\_hosts\_validos(base\_ip):

    """Verifica todos os host com a base\_ip entre 1 e 255 retorna uma lista com todos os host que tiveram resposta 0 (ativo)"""

    print("Mapeando\r")

    host\_validos = []

    return\_codes = dict()

    for i in range(1, 255):

        return\_codes[base\_ip + '{0}'.format(i)] = retorna\_codigo\_ping(base\_ip + '{0}'.format(i))

        if i %20 ==0:

            print(".", end = "")

        if return\_codes[base\_ip + '{0}'.format(i)] == 0:

            host\_validos.append(base\_ip + '{0}'.format(i))

    print("\nMapeamento pronto...")

    return host\_validos

def detalhar\_host(host\_validos):

    """Obtendo nome do host"""

    nm = nmap.PortScanner()

    for host in host\_validos:

        print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r')

        try:

            nm.scan(host)

            host\_ = Host(host, nm[host].hostname())

            #print('O IP', host, 'possui o nome', nm[host].hostname())

            for proto in nm[host].all\_protocols():

                print('----------')

                print('Protocolo : %s' % proto)

            lport = nm[host][proto].keys()

            for port in lport:

                port\_ = Port(port, nm[host][proto][port]['state'])

                host\_.ports.append(port\_)

                #print ('Porta: %s\t Estado: %s' % (port, nm[host][proto][port]['state']))

        except:

            print(':rocket: Exception')

            pass

        hosts.append(host\_)

executando = False

executou = False

ips = resumoGetNewIp()

def envolucro\_detalhar\_host():

    executando = True

    print('Iniciando coleta de dados da rede:', executando)

    meu\_ip = ips[0][1]

    print('Ip que será utilizado como base', meu\_ip)

    ip\_string = meu\_ip

    ip\_lista = ip\_string.split('.')

    base\_ip = ".".join(ip\_lista[0:3]) + '.'

    print("O teste será feito na sub rede: ", base\_ip)

    hosts\_localizados = verifica\_hosts\_validos(base\_ip)

    print ("Os host válidos são: ", hosts\_localizados)

    print('Verifica nome do host\r')

    detalhar\_host(hosts\_localizados)

    executou = True

    print('Processo finalizado', executou)

#########################################################################

def info\_processos():

    pid = subprocess.Popen('cmd.exe').pid

    p = psutil.Process(pid)

    perc\_mem = '{:.2f}'.format(p.memory\_percent())

    mem = '{:.2f}'.format(p.memory\_info().rss/1024/1024)

    tituloInfo = 'Informações sobre processos:'

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (100, 160))

    texto1 = 'Nome: ' + p.name()

    texto01 = font.render(texto1, 1, branco)

    tela.blit(texto01, (100,200))

    texto2 = 'Executável: ' + p.exe()

    texto02 = font.render(texto2, 1, branco)

    tela.blit(texto02, (100,220))

    texto3 = 'Tempo de criação: ' + time.ctime(p.create\_time())

    texto03 = font.render(texto3, 1, branco)

    tela.blit(texto03, (100,240))

    texto4 = 'Tempo de usuário: ' + str(p.cpu\_times().user) + 's'

    texto04 = font.render(texto4, 1, branco)

    tela.blit(texto04, (100,260))

    texto5 = 'Tempo de sistema: ' + str(p.cpu\_times().system) + 's'

    texto05 = font.render(texto5, 1, branco)

    tela.blit(texto05, (100,280))

    texto6 = 'Percentual de uso de CPU: ' + str(p.cpu\_percent(interval=1.0)) + '%'

    texto06 = font.render(texto6, 1, branco)

    tela.blit(texto06, (100,300))

    texto7 = 'Percentual de uso de memória: ' + perc\_mem + '%'

    texto07 = font.render(texto7, 1, branco)

    tela.blit(texto07, (100,320))

    texto8 = 'Uso de memória: ' + mem + 'MB'

    texto08 = font.render(texto8, 1, branco)

    tela.blit(texto08, (100,340))

    texto9 = 'Número de threads: ' + str(p.num\_threads())

    texto09 = font.render(texto9, 1, branco)

    tela.blit(texto09, (100,360))

#########################################################################

def get\_envolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

        if len(hosts) == 0 and not executando and not executou:

            envolucro\_detalhar\_host()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

    elif posicao == 4:

        envolucro\_arquivos()

    elif posicao == 5:

        info\_processos()

    elif posicao == 6:

        resumo()

#########################################################################

posicao\_atual = 0

#########################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

            elif event.key == pygame.K\_SPACE:

                posicao\_atual = 6

#carrossel

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < 0:

            posicao\_atual = 6

        elif posicao\_atual > 6:

            posicao\_atual = 0

        get\_envolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte v7

### Cliente:

import pygame, datetime

import socket, pickle

import os, threading

## controle aplicacao

variaveis = {

    'vermelho': (255, 0, 0),

    'azul': (29, 51, 74),

    'preto': (0, 0, 0),

    'branco': (255, 255, 255),

    'cinza': (128, 128, 128),

    'grafite': (128, 128, 128),

    'posicionamento-paginacao': (290, 530),

    'posicionamento-instrucao': (290, 560),

    'tamanho-minimo-palavra': 30,

    'porta': 9999,

    'posicao\_atual': 0,

    'pagina': 1,

    'tamanho\_tela': (800, 600),

    'diretorio\_atual': '',

    'cache\_arquivo': []

}

# inicio configuracoes pygame

#

largura\_tela = 800

altura\_tela = 600

terminou = False

count = 60

variaveis['diretorio\_atual'] = os.getcwd()

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Projeto de bloco - Carlos Henrique")

pygame.display.init()

pygame.font.init()

clock = pygame.time.Clock()

font = pygame.font.SysFont("arial", 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

#

# fim configuracoes pygame

# inicio thread

#

class ThreadLog(threading.Thread):

    def \_\_init\_\_(self, threadID, name, counter):

        threading.Thread.\_\_init\_\_(self)

        self.threadID = threadID

        self.name = name

        self.counter = counter

    def run(self):

        print('ThreadLog >> iniciando escrita do log')

        while True:

            if len(variaveis['cache\_arquivo'])> 1:

                logs = variaveis['cache\_arquivo']

                log\_aux = logs[len(logs) - 1]

                set\_log(log\_aux)

                del(logs[len(logs) - 1])

t = ThreadLog(1, 'ThreadLog', 2)

t.start()

#

# fim thread

# inicio socket

#

socket\_ = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

socket\_.connect((socket.gethostname(), 9999))

def request(message):

    socket\_.send(message.encode('ascii'))

    received = socket\_.recv(90000)

    response = pickle.loads(received)

    return response

#

# fim socket

# inicio exibir informações em tela

#

def get\_envolucro\_cpu():

    log\_ = 'REQUEST: > ' + str(datetime.datetime.now()) + ' > cpu'

    variaveis['cache\_arquivo'].append(log\_)

    response\_cpu = request('cpu')

    log\_ = 'RESPONSE: > ' + str(datetime.datetime.now()) + ' > ' + str(response\_cpu)

    variaveis['cache\_arquivo'].append(log\_)

    set\_info\_cpu(response\_cpu)

    set\_grafico\_cpu(superficie\_info\_cpu, response\_cpu)

def get\_envolucro\_memoria():

    log\_ = 'REQUEST: > ' + str(datetime.datetime.now()) + ' > memoria'

    variaveis['cache\_arquivo'].append(log\_)

    response\_memoria = request('memoria')

    log\_ = 'RESPONSE: > ' + str(datetime.datetime.now()) + ' > ' + str(response\_memoria)

    variaveis['cache\_arquivo'].append(log\_)

    set\_info\_memoria(response\_memoria)

    set\_grafico\_memoria(response\_memoria)

def get\_envolucro\_disco():

    log\_ = 'REQUEST: > ' + str(datetime.datetime.now()) + ' > disco'

    variaveis['cache\_arquivo'].append(log\_)

    response\_disco = request('disco')

    log\_ = 'RESPONSE: > ' + str(datetime.datetime.now()) + ' > ' + str(response\_disco)

    variaveis['cache\_arquivo'].append(log\_)

    set\_info\_disco(response\_disco)

    set\_grafico\_disco(response\_disco)

def get\_envolucro\_rede():

    log\_ = 'REQUEST: > ' + str(datetime.datetime.now()) + ' > ips'

    variaveis['cache\_arquivo'].append(log\_)

    response\_ips = request('ips')

    log\_ = 'RESPONSE: > ' + str(datetime.datetime.now()) + ' > ' + str(response\_ips)

    variaveis['cache\_arquivo'].append(log\_)

    log\_ = 'REQUEST: > ' + str(datetime.datetime.now()) + ' > trafego'

    variaveis['cache\_arquivo'].append(log\_)

    response\_trafego = request('trafego')

    log\_ = 'RESPONSE: > ' + str(datetime.datetime.now()) + ' > ' + str(response\_trafego)

    variaveis['cache\_arquivo'].append(log\_)

    log\_ = 'REQUEST: > ', str(datetime.datetime.now()), ' > ' , ' rede'

    variaveis['cache\_arquivo'].append(log\_)

    response\_host = request('rede')

    log\_ = 'RESPONSE: > ' + str(datetime.datetime.now()) + ' > ' + str(response\_host)

    variaveis['cache\_arquivo'].append(log\_)

    set\_info\_rede(response\_ips, response\_trafego, response\_host)

    set\_info\_hosts\_rede(response\_host)

response\_cache\_arquivo = ''

def get\_envolucro\_arquivo():

    global response\_cache\_arquivo

    if response\_cache\_arquivo != '' and int(variaveis['pagina']) > int(response\_cache\_arquivo[2]['total\_paginas']):

        variaveis['pagina'] = int(response\_cache\_arquivo[1]['total\_paginas'])

    log\_ = 'REQUEST: > ' + str(datetime.datetime.now()) + ' > arquivos/' + str(variaveis['pagina'])

    variaveis['cache\_arquivo'].append(log\_)

    response\_arquivos = request('arquivos/' + str(variaveis['pagina']))

    response\_cache\_arquivo = response\_arquivos

    log\_ = 'RESPONSE: > ' + str(datetime.datetime.now()) + ' > ' + str(response\_arquivos)

    variaveis['cache\_arquivo'].append(log\_)

    set\_info\_arquivo(response\_arquivos)

response\_cache\_processo = ''

def get\_envolucro\_processos():

    global response\_cache\_processo

    if response\_cache\_processo != '' and int(variaveis['pagina']) > int(response\_cache\_processo['total\_paginas']):

        variaveis['pagina'] = int(response\_cache\_processo['total\_paginas'])

    log\_ = 'REQUEST: > ' + str(datetime.datetime.now()) + ' > processo/' + str(variaveis['pagina'])

    variaveis['cache\_arquivo'].append(log\_)

    response\_processos = request('processo/' + str(variaveis['pagina']))

    response\_cache\_processo = response\_processos

    log\_ = 'RESPONSE: > ' + str(datetime.datetime.now()) + ' > ' + str(response\_processos)

    variaveis['cache\_arquivo'].append(log\_)

    set\_info\_processo(response\_processos)

def get\_envolucro\_resumo():

    log\_ ='REQUEST: > ' + str(datetime.datetime.now()) + ' > resumo'

    variaveis['cache\_arquivo'].append(log\_)

    response\_resumo = request('resumo')

    log\_ ='RESPONSE: > ' + str(datetime.datetime.now()) + ' > ' + str(response\_resumo)

    variaveis['cache\_arquivo'].append(log\_)

    set\_info\_resumo(response\_resumo)

def get\_envolucro(posicao):

    if posicao == 0:

        get\_envolucro\_cpu()

    elif posicao == 1:

        get\_envolucro\_memoria()

    elif posicao == 2:

        get\_envolucro\_disco()

    elif posicao == 3:

        get\_envolucro\_rede()

    elif posicao == 4:

        get\_envolucro\_arquivo()

    elif posicao == 5:

        get\_envolucro\_processos()

    elif posicao == 6:

        get\_envolucro\_resumo()

def set\_info\_cpu(cpu):

    superficie\_info\_cpu.fill(variaveis['grafite'])

    # label

    text\_arquitetura = font.render('Arquitetura:', True, (30, 0, 0))

    superficie\_info\_cpu.blit(text\_arquitetura, (40, 30))

    # valor

    valor\_arquitetura = font.render(cpu['arquitetura'], True, (30, 0, 0))

    superficie\_info\_cpu.blit(valor\_arquitetura, (180, 30))

    # label

    text\_bits = font.render('Total Bits:', True, (30, 0, 0))

    superficie\_info\_cpu.blit(text\_bits, (40, 50))

    # valor

    valor\_bits = font.render(cpu['bits'], True, (30, 0, 0))

    superficie\_info\_cpu.blit(valor\_bits, (180, 50))

    # label

    text\_frequencia = font.render('Frequência:', True, (30, 0, 0))

    superficie\_info\_cpu.blit(text\_frequencia, (40, 70))

    # valor

    valor\_frquencia = font.render(cpu['frequencia'], True, (30, 0, 0))

    superficie\_info\_cpu.blit(valor\_frquencia, (180, 70))

    # label

    text\_nucleo = font.render('Núcleos (físico):', True, (30, 0, 0))

    superficie\_info\_cpu.blit(text\_nucleo, (40, 90))

    # valor

    valor\_nucleo = font.render(cpu['nucleos'], True, (30, 0, 0))

    superficie\_info\_cpu.blit(valor\_nucleo, (180, 90))

    # label

    texto\_nome = font.render('Nome:', True, variaveis['preto'])

    superficie\_info\_cpu.blit(texto\_nome, (40, 110))

    # valor

    valor\_nome = font.render(cpu['nome'], True, variaveis['preto'])

    superficie\_info\_cpu.blit(valor\_nome, (180, 110))

    tela.blit(superficie\_info\_cpu, (0,0))

def set\_grafico\_cpu(s, cpu):

    l\_cpu\_percent = cpu['l\_cpu\_percent']

    s.fill(variaveis['grafite'])

    capacidade = cpu['capacidade']

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width() - 2 \* y - (num\_cpu + 1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, variaveis['azul'], (d, y + 20, larg, alt))

        pygame.draw.rect(s, variaveis['branco'], (d, y + 20, larg, (1 - i / 100) \* alt))

        d = d + larg + desl

    text = font.render("Uso de CPU por núcleo total: " + str(capacidade) + "%", 1, variaveis['branco'])

    s.blit(text, (20, 5))

    # instrucao navegacao

    instrucao = font.render('Tecle ← ou → para navegar', True, variaveis['preto'])

    tela.blit(instrucao, variaveis['posicionamento-instrucao'])

    tela.blit(s, (0, 300))

def set\_info\_memoria(memoria):

    tela.fill(variaveis['grafite'])

    titulo = font.render("\*\* Informações de Memória \*\*" , 1, variaveis['azul'])

    tela.blit(titulo, (15, 10))

    # titulo

    texto = font.render('Capacidade', True, variaveis['preto'])

    tela.blit(texto, (15, 60))

    # valor

    tela.blit(font.render(str(memoria['capacidade']) + 'GB', True, variaveis['preto']), (155, 60))

    #titulo

    texto = font.render('Disponível', True, variaveis['preto'])

    tela.blit(texto, (15, 80))

    # valor

    tela.blit(font.render(str(memoria['disponivel']), True, variaveis['preto']), (155, 80))

def set\_grafico\_memoria(memoria):

    memoria = memoria['memoria']

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(tela, variaveis['branco'], (15, 270, largura, 5))

    largura = largura \* memoria.percent / 100

    pygame.draw.rect(tela, variaveis['azul'], (15, 270, largura, 5))

    total = round(memoria.total / (1024 \* 1024 \* 1024), 2)

    porcentagem = memoria.percent

    texto\_da\_barra = ('Percentual usado: {}% (Total: {} GB)'.format(porcentagem, total))

    text = font.render(texto\_da\_barra, 1, variaveis['branco'])

    tela.blit(text, (20, 240))

    # instrucao navegacao

    instrucao = font.render('Tecle ← ou → para navegar', True, variaveis['preto'])

    tela.blit(instrucao, variaveis['posicionamento-instrucao'])

def set\_info\_disco(memoria):

    tela.fill(variaveis['grafite'])

    titulo = font.render("\*\* Informações do Disco \*\*" , 1, variaveis['azul'])

    tela.blit(titulo, (15, 10))

    # titulo

    texto = font.render('Capacidade', True, variaveis['preto'])

    tela.blit(texto, (15, 60))

    # valor

    tela.blit(font.render(str(memoria['total']) + ' GB', True, variaveis['preto']), (155, 60))

    #titulo

    texto = font.render('Disponível', True, variaveis['preto'])

    tela.blit(texto, (15, 80))

    # valor

    tela.blit(font.render(str(memoria['livre']) + ' GB', True, variaveis['preto']), (155, 80))

    #titulo

    texto = font.render('Usado', True, variaveis['preto'])

    tela.blit(texto, (15, 100))

    # valor

    tela.blit(font.render(str(memoria['usado']) + ' GB', True, variaveis['preto']), (155, 100))

def set\_grafico\_disco(memoria):

    disco\_aux = memoria['disco']

    total = memoria['total']

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(tela, variaveis['branco'], (15, 270, largura, 5))

    consumo = (largura \* disco\_aux.percent) / 100

    pygame.draw.rect(tela, variaveis['azul'], (15, 270, consumo, 5))

    texto\_da\_barra = ('Uso de Disco: {}% (Total: {} GB)'.format(disco\_aux.percent, total))

    texto = font.render(texto\_da\_barra, 1, variaveis['branco'])

    tela.blit(texto, (20, 240))

    # instrucao navegacao

    instrucao = font.render('Tecle ← ou → para navegar', True, variaveis['preto'])

    tela.blit(instrucao, variaveis['posicionamento-instrucao'])

def set\_info\_rede(ips, trafegos, hosts):

    tela.fill(variaveis['grafite'])

    titulo = font.render("\*\* Informações de Rede \*\*" , 1, variaveis['azul'])

    tela.blit(titulo, (15, 10))

    titulo = font.render("Interface                              IP                        Mascara             Pct. Enviado       Pct. Recebido" , 1, variaveis['preto'])

    tela.blit(titulo, (15, 75))

    espacos = 100

    for host in ips:

        interface = host[0]

        trafego\_da\_interface = get\_trafego\_da\_interface(interface, trafegos)

        ip = str(host[1])

        if ip !='127.0.0.1':

            pct\_recebido = size\_format(trafego\_da\_interface['pacotes\_recebidos'])

            pct\_enviado = size\_format(trafego\_da\_interface['pacotes\_enviados'])

            pct\_enviado\_formatado = '{:^30}'.format(str(pct\_enviado))

            pct\_recebido\_formatado = '{:^15}'.format(str(pct\_recebido))

            nome\_interface\_formatada = get\_nova\_string(str(host[0]))

            ip\_formatada = str(host[1])

            ip\_formatada\_ = '{:<20}'.format(ip\_formatada)

            mascara = str(host[2])

            mascara\_formatada = '{:^15}'.format(mascara)

            texto = font.render(nome\_interface\_formatada + ip\_formatada\_ +  mascara\_formatada + pct\_enviado\_formatado + pct\_recebido\_formatado, 1, variaveis['preto'])

            tela.blit(texto, (15, espacos))

            espacos += 25

    # exibir msg de informacao: escaneando rede

    if hosts == 'NoNe':

        texto\_atencao = font.render('Lendo dados da rede. Aguarde...', 10, variaveis['vermelho'])

        tela.blit(texto\_atencao, (260, 185))

def set\_info\_hosts\_rede(hosts):

    espacos = 300

    if hosts != 'NoNe':

        for host in hosts:

            host\_name = ""

            if host['nome'] != "":

                host\_name = host['nome']

            else:

                host\_name = "NÃO IDENTIFICADO"

            cor = ""

            if host\_name == "NÃO IDENTIFICADO":

                cor = variaveis['vermelho']

            else:

                cor = variaveis['azul']

            texto = font.render(host['ip'] + ': Nome: ' + host\_name, 1, variaveis['azul'])

            tela.blit(texto, (15, espacos + 5))

            espacos += 15

            for porta in host['portas']:

                porta\_label = font.render("Porta: ", 1, variaveis['branco'])

                tela.blit(porta\_label, (15, espacos + 10))

                porta\_text = font.render(str(porta['porta']), 1, variaveis['branco'])

                tela.blit(porta\_text, (70, espacos + 10))

                estado\_label = font.render("Estado: ", 1, variaveis['branco'])

                tela.blit(estado\_label, (140, espacos + 10))

                estado = font.render(porta['estado'], 1, variaveis['branco'])

                tela.blit(estado, (210, espacos + 10))

                espacos += 15

            espacos += 20

    # instrucao navegacao

    instrucao = font.render('Tecle ← ou → para navegar', True, variaveis['preto'])

    tela.blit(instrucao, variaveis['posicionamento-instrucao'])

def set\_info\_arquivo(response):

    tela.fill(variaveis['grafite'])

    titulo = font.render("\*\* Arquivos do diretório \*\*" , 1, variaveis['azul'])

    tela.blit(titulo, (15, 10))

    diretorio = font.render("> " + response[0], 1, variaveis['preto'])

    tela.blit(diretorio, (15, 45))

    titulo = font.render("Nome                                               Data Criação                      Data Modificação             Tamanho" , 1, variaveis['preto'])

    tela.blit(titulo, (15, 75))

    espacos = 100

    tempo\_execucao = response[1]

    arquivos = response[2]

    total\_paginas = arquivos['total\_paginas']

    pagina\_atual = arquivos['pagina\_atual']

    for arquivo in arquivos['elementos']:

        tamanho\_arquivo = size\_format(arquivo['tamanho'])

        nome\_arquivo = get\_nova\_string(arquivo['nome'])

        texto\_formatado = font.render(nome\_arquivo , 1, variaveis['preto'])

        tela.blit(texto\_formatado, (15, espacos))

        data\_criacao = datetime.datetime.fromtimestamp(arquivo['data\_criacao']).strftime("%d-%m-%Y %H:%M:%S")

        texto\_formatado = font.render(data\_criacao , 1, variaveis['preto'])

        tela.blit(texto\_formatado, (300, espacos))

        data\_modificacao = datetime.datetime.fromtimestamp(arquivo['data\_modificacao']).strftime("%d-%m-%Y %H:%M:%S")

        data\_modificacao\_formatado = font.render(data\_modificacao , 1, variaveis['preto'])

        tela.blit(data\_modificacao\_formatado, (500, espacos))

        tamanho\_arquivo\_formatado = font.render(tamanho\_arquivo, 1, variaveis['preto'])

        tela.blit(tamanho\_arquivo\_formatado, (700, espacos))

        espacos += 25

    get\_paginar(total\_paginas, pagina\_atual)

    informacao = font.render(tempo\_execucao[0], True, variaveis['branco'])

    tela.blit(informacao, (15, 480))

    informacao = font.render(tempo\_execucao[1], True, variaveis['branco'])

    tela.blit(informacao, (15, 500))

    # instrucao navegacao

    instrucao = font.render('Tecle ← ou → para navegar', True, variaveis['preto'])

    tela.blit(instrucao, variaveis['posicionamento-instrucao'])

def set\_info\_processo(response):

    total\_de\_paginas = response['total\_paginas']

    pagina\_atual = response['pagina\_atual']

    tela.fill(variaveis['grafite'])

    titulo = font.render("\*\* Lista dos processos em execução \*\*" , 1, variaveis['azul'])

    tela.blit(titulo, (15, 10))

    titulo = font.render("    PID        % Uso        Mem. Usada     Threads Usada            Tempo                      Nome" , 1, variaveis['preto'])

    tela.blit(titulo, (15, 55))

    espacos = 100

    processos = response['elementos']

    for processo in processos:

        text\_pid = '{:<15}'.format(str(processo['pid']))

        texto = font.render(text\_pid, 1, variaveis['preto'])

        tela.blit(texto, (40, espacos))

        text\_percentual\_uso = '{:<20}'.format(str(format(processo['percentual\_uso'], '.2f')))

        texto = font.render(text\_percentual\_uso, 1, variaveis['preto'])

        tela.blit(texto, (110, espacos))

        text\_memoria\_usada = '{:<20}'.format(size\_format(processo['memoria\_usada']))

        texto = font.render(text\_memoria\_usada, 1, variaveis['preto'])

        tela.blit(texto, (220, espacos))

        text\_threads\_processo = '{:<20}'.format(str(format(processo['threads\_processo'], '.2f')))

        texto = font.render(text\_threads\_processo, 1, variaveis['preto'])

        tela.blit(texto, (350, espacos))

        texto\_tempo\_exec = '{:<20}'.format(processo['tempo\_usuario'])

        texto = font.render(texto\_tempo\_exec, 1, variaveis['preto'])

        tela.blit(texto, (490, espacos))

        text\_nome = '{:<30}'.format(processo['nome'])

        texto = font.render(text\_nome, 1, variaveis['preto'])

        tela.blit(texto, (620, espacos))

        espacos += 25

    get\_paginar(total\_de\_paginas, pagina\_atual)

    # instrucao navegacao

    instrucao = font.render('Tecle ← ou → para navegar', True, variaveis['preto'])

    tela.blit(instrucao, variaveis['posicionamento-instrucao'])

def set\_info\_resumo(response):

    tela.fill(variaveis['grafite'])

    titulo = font.render("\*\* Resumo dos dados coletados \*\*" , 1, variaveis['azul'])

    tela.blit(titulo, (15, 10))

    titulo = font.render("CPU" , 1, variaveis['azul'])

    tela.blit(titulo, (15, 60))

    processador = response['cpu']

    disco = response['disco']

    memoria\_total = response['memoria\_capacidade']

    memoria\_disponivel = response['memoria\_disponivel']

    memoria\_usada = memoria\_total - memoria\_disponivel

    # obtem o ip do usuario

    host = response['ips'][0][1]

    if host == '127.0.0.1':

        host = response['ips'][1][1]

    # titulo

    texto = font.render('Processador:', True, variaveis['preto'])

    tela.blit(texto, (15, 80))

    # valor

    tela.blit(font.render(processador['nome'], True, variaveis['preto']), (155, 80))

    # titulo

    texto = font.render('Frequência:', True, variaveis['preto'])

    tela.blit(texto, (15, 100))

    # valor

    tela.blit(font.render(processador['frequencia'], True, variaveis['preto']), (155, 100))

    # titulo

    texto = font.render('Bits:', True, variaveis['preto'])

    tela.blit(texto, (15, 120))

    # valor

    tela.blit(font.render(processador['bits'], True, variaveis['preto']), (155, 120))

    #

    tela.blit(font.render('----------------------------------------------------------', True, variaveis['branco']), (180, 150))

    titulo = font.render("Disco" , 1, variaveis['azul'])

    tela.blit(titulo, (15, 170))

    # titulo

    texto = font.render('Total:', True, variaveis['preto'])

    tela.blit(texto, (15, 190))

    # valor

    tela.blit(font.render(size\_format(disco['disco'][0]), True, variaveis['preto']), (155, 190))

    # titulo

    texto = font.render('Livre:', True, variaveis['preto'])

    tela.blit(texto, (15, 210))

    # valor

    tela.blit(font.render(size\_format(disco['disco'][2]), True, variaveis['preto']), (155, 210))

    # titulo

    texto = font.render('Usado:', True, variaveis['preto'])

    tela.blit(texto, (15, 230))

    # valor

    tela.blit(font.render(size\_format(disco['disco'][1]), True, variaveis['preto']), (155, 230))

    #

    tela.blit(font.render('----------------------------------------------------------', True, variaveis['branco']), (180, 260))

    titulo = font.render("Memória" , 1, variaveis['azul'])

    tela.blit(titulo, (15, 280))

    # titulo

    texto = font.render('Total:', True, variaveis['preto'])

    tela.blit(texto, (15, 300))

    # valor

    tela.blit(font.render(str(memoria\_total) + 'GB', True, variaveis['preto']), (155, 300))

    # titulo

    texto = font.render('Livre:', True, variaveis['preto'])

    tela.blit(texto, (15, 320))

    # valor

    tela.blit(font.render(str(memoria\_disponivel) + 'GB', True, variaveis['preto']), (155, 320))

    # titulo

    texto = font.render('Usado:', True, variaveis['preto'])

    tela.blit(texto, (15, 340))

    # valor

    tela.blit(font.render(str((format(memoria\_usada, '.2f'))) + 'GB', True, variaveis['preto']), (155, 340))

    #

    tela.blit(font.render('----------------------------------------------------------', True, variaveis['branco']), (180, 370))

    titulo = font.render("Rede" , 1, variaveis['azul'])

    tela.blit(titulo, (15, 390))

    # titulo

    texto = font.render('IP:', True, variaveis['preto'])

    tela.blit(texto, (15, 410))

    # valor

    tela.blit(font.render(host, True, variaveis['preto']), (155, 410))

    # instrucao navegacao

    instrucao = font.render('Tecle ← ou → para navegar', True, variaveis['preto'])

    tela.blit(instrucao, variaveis['posicionamento-instrucao'])

def get\_trafego\_da\_interface(interface, trafegos):

    response = ''

    medicoes = trafegos[len(trafegos) - 1]

    for trafego in medicoes:

        if trafego['interface'] == interface:

            response = trafego

            break

    return response

def get\_nova\_string(palavra):

    palavra\_aux = palavra

    tamanho\_minimo = variaveis['tamanho-minimo-palavra']

    tamanho\_palavra = len(palavra\_aux)

    if tamanho\_palavra > tamanho\_minimo:

        # recorta a string

        palavra\_aux = '{:.30}'.format(palavra)

    else:

        # adiciona espacos

        while tamanho\_palavra != tamanho\_minimo:

            palavra\_aux = palavra\_aux + " "

            tamanho\_palavra = len(palavra\_aux)

    return palavra\_aux

def size\_format(b):

    if b < 1000:

              return '%i' % b + 'B'

    elif 1000 <= b < 1000000:

        return '%.1f' % float(b/1000) + 'KB'

    elif 1000000 <= b < 1000000000:

        return '%.1f' % float(b/1000000) + 'MB'

    elif 1000000000 <= b < 1000000000000:

        return '%.1f' % float(b/1000000000) + 'GB'

    elif 1000000000000 <= b:

        return '%.1f' % float(b/1000000000000) + 'TB'

def get\_paginar(total\_paginas, pagina\_atual):

    count\_1 = 1

    count\_2 = 1

    count\_3 = 1

    count\_4 = 1

    # instrucao paginacao

    if int(total\_paginas) > 1:

        instrucao = font.render('Tecle + ou - para paginar', True, variaveis['azul'])

        tela.blit(instrucao, variaveis['posicionamento-paginacao'])

    for n in range(1, total\_paginas + 1):

        cor = variaveis['branco']

        if n == int(pagina\_atual):

            cor = variaveis['azul']

        if n <= 20:

            area = (30 \* n, 300, 25, 25)

            pygame.draw.rect(tela, cor, area)

            instrucao = font.render(str(n), True, variaveis['preto'])

            tela.blit(instrucao, ((30 \* n) + 5, 300))

        elif n > 20 and n <= 40:

            area = (30 \* count\_1, 330, 25, 25)

            count\_2 = 1

            pygame.draw.rect(tela, cor, area)

            instrucao = font.render(str(n), True, variaveis['preto'])

            tela.blit(instrucao, ((30 \* count\_1) + 5, 330))

            count\_2 = 1

            count\_1 = count\_1 + 1

            count\_2 = 1

        elif n > 40 and n <=60:

            area = (30 \* count\_2, 360, 25, 25)

            pygame.draw.rect(tela, cor, area)

            instrucao = font.render(str(n), True, variaveis['preto'])

            tela.blit(instrucao, ((30 \* count\_2) + 5, 360))

            count\_2 = count\_2 + 1

        elif n > 60 and n <=80:

            area = (30 \* count\_3, 390, 25, 25)

            pygame.draw.rect(tela, cor, area)

            instrucao = font.render(str(n), True, variaveis['preto'])

            tela.blit(instrucao, ((30 \* count\_3) + 5, 390))

            count\_3 = count\_3 + 1

        elif n > 80 and n <= 100:

            area = (30 \* count\_4, 420, 25, 25)

            pygame.draw.rect(tela, cor, area)

            instrucao = font.render(str(n), True, variaveis['preto'])

            tela.blit(instrucao, ((30 \* count\_4) + 5, 420))

            count\_4 = count\_4 + 1

def set\_log(message):

    try:

        file = open(variaveis['diretorio\_atual'] + '\\log-request.txt', 'a')

        file.write(str(message) + '\n')

        file.close()

    except:

        print('ERROR >>> SEM PERMISSÃO PARA ABRIR ARQUIVO')

#

#fim exibir informações em tela

while not terminou:

    # monitorando eventos

    for event in pygame.event.get():

        # para a aplicacao

        if event.type == pygame.QUIT:

            terminou = True

        # monitora interação do usuario

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                variaveis['pagina'] = 1

                variaveis['posicao\_atual'] = variaveis['posicao\_atual'] + 1

            elif event.key == pygame.K\_LEFT:

                variaveis['pagina'] = 1

                variaveis['posicao\_atual'] = variaveis['posicao\_atual'] - 1

            elif event.key == pygame.K\_SPACE:

                variaveis['posicao\_atual'] = 6

            elif event.key == pygame.K\_KP\_PLUS:

                variaveis['pagina'] = variaveis['pagina'] + 1

            elif event.key == pygame.K\_KP\_MINUS:

                if variaveis['pagina'] > 1:

                    variaveis['pagina'] = variaveis['pagina'] - 1

                else:

                    variaveis['pagina'] = 1

#carrossel

    if count == 60:

        tela.fill(variaveis['grafite'])

        if variaveis['posicao\_atual'] < 0:

            variaveis['posicao\_atual'] = 6

        elif variaveis['posicao\_atual'] > 6:

            variaveis['posicao\_atual'] = 0

        get\_envolucro(variaveis['posicao\_atual'])

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

### servidor:

import socket, psutil, pickle, cpuinfo, threading

import time, sched, os, platform, subprocess

import nmap, math

# controle aplicacao

variaveis = {

    'cpu': [],

    'memoria': [],

    'disco':[],

    'processo': [],

    'arquivos': [],

    'sched':[],

    'ips': [],

    'hosts\_detalhado': [],

    'trafego' : [],

    'total\_elementos\_por\_pagina': 5

}

# configuracao processador

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

# inicio classes

class Host:

    def \_\_init\_\_(self, ip, name):

        self.ip = ip

        self.name = name

        self.ports = []

class Porta:

    def \_\_init\_\_(self, port, state):

        self.port = port

        self.state = state

class Processo:

    def \_\_init\_\_(self, pid, nome, percentual\_uso, memoria\_usada, threads\_processo, tempo\_usuario, data\_criacao):

        self.pid = pid

        self.nome = nome

        self.percentual\_uso = percentual\_uso

        self.memoria\_usada = memoria\_usada

        self.threads\_processo = threads\_processo

        self.tempo\_usuario = tempo\_usuario

        self.data\_criacao = data\_criacao

    def to\_map(self):

        return { 'pid': self.pid, 'nome' : self.nome, 'percentual\_uso' : self.percentual\_uso, 'memoria\_usada' : self.memoria\_usada, 'threads\_processo' : self.threads\_processo, 'tempo\_usuario' : self.tempo\_usuario, 'data\_criacao' : self.data\_criacao }

class Arquivo:

    def \_\_init\_\_(self, nome, tamanho, data\_criacao, data\_modificacao):

        self.nome = nome

        self.tamanho = tamanho

        self.data\_criacao = data\_criacao

        self.data\_modificacao = data\_modificacao

    def to\_map(self):

        return {

            'nome': self.nome

        ,   'tamanho': self.tamanho

        ,   'data\_criacao': self.data\_criacao

        ,   'data\_modificacao': self.data\_modificacao

        #,   'diretorio': self.diretorio

    }

class CPU():

    def \_\_init\_\_(self, nome, arquitetura, bits, frequencia, nucleos, l\_cpu\_percent, capacidade, num\_cpu):

        self.nome = nome

        self.arquitetura = arquitetura

        self.bits = bits

        self.frequencia = frequencia

        self.nucleos = nucleos

        self.l\_cpu\_percent = l\_cpu\_percent

        self.capacidade = capacidade

        self.num\_cpu = num\_cpu

    def to\_map(self):

        return { 'nome': self.nome, 'arquitetura': self.arquitetura, 'bits': self.bits, 'frequencia': self.frequencia, 'nucleos': self.nucleos, 'l\_cpu\_percent': self.l\_cpu\_percent, 'capacidade': self.capacidade, 'num\_cpu': self.num\_cpu}

class Memoria():

    def \_\_init\_\_(self, memoria, capacidade, disponivel):

        self.memoria = memoria

        self.capacidade = capacidade

        self.disponivel = disponivel

    def get\_map(self):

        return { 'memoria': self.memoria, 'capacidade' : self.capacidade, 'disponivel' : self.disponivel}

class Disco():

    def \_\_init\_\_(self, disco, usado, total, livre):

        self.disco = disco

        self.usado = usado

        self.total = total

        self.livre = livre

    def to\_map(self):

        return {'disco': self.disco, 'usado': self.usado, 'total': self.total, 'livre': self.livre}

class Trafego():

    def \_\_init\_\_(self, interface, enviados, recebidos, pacotes\_enviados, pacotes\_recebidos):

        self.interface = interface

        self.enviados = enviados

        self.recebidos = recebidos

        self.pacotes\_enviados = pacotes\_enviados

        self.pacotes\_recebidos = pacotes\_recebidos

    def to\_map(self):

        return { 'interface' : self.interface, 'enviados': self.enviados, 'recebidos' : self.recebidos, 'pacotes\_enviados' : self.pacotes\_enviados, 'pacotes\_recebidos': self.pacotes\_recebidos }

class Resumo():

    def \_\_init\_\_(self, total\_processos, memoria\_capacidade, memoria\_disponivel, cpu, disco):

        self.total\_processos = total\_processos

        self.memoria\_capacidade = memoria\_capacidade

        self.memoria\_disponivel = memoria\_disponivel

        self.ips = []

        self.cpu = cpu

        self.disco = disco

    def to\_map(self):

        return {

            'total\_processos': self.total\_processos,

            'ips': self.ips,

            'memoria\_capacidade' : self.memoria\_capacidade,

            'memoria\_disponivel': self.memoria\_disponivel,

            'cpu': self.cpu,

            'disco': self.disco

        }

# fim classes

# inicio threads

class ThreadIps(threading.Thread):

    def \_\_init\_\_(self, threadID, name, counter):

        threading.Thread.\_\_init\_\_(self)

        self.threadID = threadID

        self.name = name

        self.counter = counter

    def run(self):

        while True:

            if len(variaveis['ips']) == 0:

                print ("Starting ThreadIps" + self.name)

                variaveis['ips'] = get\_meus\_ips()

                print ("Exiting ThreadIps" + self.name)

                time.sleep(30)

            else:

                break

class ThreadRede(threading.Thread):

   def \_\_init\_\_(self, threadID, name, counter):

      threading.Thread.\_\_init\_\_(self)

      self.threadID = threadID

      self.name = name

      self.counter = counter

   def run(self):

       while True:

           print ("Starting scanning rede" + self.name)

           get\_hosts()

           print ("Exiting scanning rede" + self.name)

           time.sleep(20)

class ThreadDisco(threading.Thread):

    def \_\_init\_\_(self, threadID, name, counter):

        threading.Thread.\_\_init\_\_(self)

        self.threadID = threadID

        self.name = name

        self.counter = counter

    def run(self):

        while True:

            print ("Starting thread" + self.name)

            get\_info\_disco()

            print ("Sleeping..." + self.name)

            time.sleep(20)

class ThreadCpu(threading.Thread):

    def \_\_init\_\_(self, threadID, name, counter):

        threading.Thread.\_\_init\_\_(self)

        self.threadID = threadID

        self.name = name

        self.counter = counter

    def run(self):

        while True:

            print ("Starting thread" + self.name)

            get\_info\_cpu()

            print ("Sleeping..." + self.name)

            time.sleep(20)

class ThreadSched(threading.Thread):

    def \_\_init\_\_(self, threadID, name, counter):

        threading.Thread.\_\_init\_\_(self)

        self.threadID = threadID

        self.name = name

        self.counter = counter

    def run(self):

        while True:

            print ("Starting thread" + self.name)

            get\_shed\_sheduler\_arquivos()

            print ("Sleeping..." + self.name)

            time.sleep(20)

class ThreadTrafegoRede(threading.Thread):

    def \_\_init\_\_(self, threadID, name, counter):

        threading.Thread.\_\_init\_\_(self)

        self.threadID = threadID

        self.name = name

        self.counter = counter

    def run(self):

        while True:

            print ("Starting thread" + self.name)

            get\_trafego\_host()

            print ("Sleeping..." + self.name)

            time.sleep(2)

class ThreadMemoria(threading.Thread):

    def \_\_init\_\_(self, threadID, name, counter):

        threading.Thread.\_\_init\_\_(self)

        self.threadID = threadID

        self.name = name

        self.counter = counter

    def run(self):

        while True:

            print ("Starting thread" + self.name)

            get\_info\_memoria()

            print ("Sleeping..." + self.name)

            time.sleep(10)

class ThreadProcesso(threading.Thread):

    def \_\_init\_\_(self, threadID, name, counter):

        threading.Thread.\_\_init\_\_(self)

        self.threadID = threadID

        self.name = name

        self.counter = counter

    def run(self):

        while True:

            print ("Starting thread" + self.name)

            aux = get\_processo()

            variaveis['processo'].clear

            variaveis['processo'] = aux

            print ("Sleeping..." + self.name)

            time.sleep(10)

# fim threads

# inicia funcoes

def get\_info\_disco():

    """ RESPONSAVEL POR OBTER AS INFORMACOES DO DISCO """

    disco = psutil.disk\_usage('.')

    usado = round((disco.total - disco.free)  / 1024\*\*3, 2)

    total = round(disco.total / (1024\*\*3), 2)

    livre = round(disco.free/(1024\*\*3),2)

    disco\_aux = Disco(disco, usado, total, livre)

    variaveis['disco'].append(disco\_aux)

def get\_info\_cpu():

    """ RESPONSAVEL POR OBTER AS INFORMACOES DA CPU """

    nome\_cpu = str(info\_cpu['brand\_raw'])

    arquitetura\_cpu = str(info\_cpu['arch'])

    bits\_cpu = str(info\_cpu['bits'])

    frq\_cpu = str(info\_cpu['hz\_actual\_friendly'])

    nucleos = str(info\_cpu['count'])

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    capacidade = psutil.cpu\_percent(interval=1)

    num\_cpu = len(l\_cpu\_percent)

    cpu = CPU(nome\_cpu, arquitetura\_cpu, bits\_cpu, frq\_cpu, nucleos, l\_cpu\_percent, capacidade, num\_cpu)

    variaveis['cpu'].append(cpu)

def get\_arquivos():

    """ RESPONSAVEL POR OBTER OS ARQUIVOS """

    arquivos = os.listdir()

    diretorio = os.getcwd()

    arquivo\_response = []

    for arquivo in arquivos:

        tamanho = os.stat(arquivo).st\_size

        criacao = os.stat(arquivo).st\_ctime

        modificacao = os.stat(arquivo).st\_mtime

        arquivo\_aux = Arquivo(arquivo, tamanho, criacao, modificacao)

        arquivo\_response.append(arquivo\_aux.to\_map())

    variaveis['arquivos'].clear()

    variaveis['arquivos'].append(arquivo\_response)

def get\_shed\_sheduler\_arquivos():

    """ RESPONSAVEL POR OBTER TEMPO DA OBTENCAO DE ARQUIVOS """

    inicio = time.time()

    inicioClock = time.process\_time()

    sched\_ = sched.scheduler(time.time, time.sleep)

    sched\_.enter(3, 1, get\_arquivos())

    tempoFinal = 'TEMPO FINAL: %s | CLOCK FINAL: %0.2f' % (time.ctime(), time.process\_time())

    final = time.time() - inicio

    finalClock = time.process\_time() - inicioClock

    tempoUsado = 'TEMPO USADO NESSA CHAMADA: %0.3f segundos | CLOCK USADO NESSA CHAMADA: %0.2f' % (final, finalClock)

    variaveis['sched'].append((tempoFinal, tempoUsado))

def getNewIp(sistema):

    for interface, snics in psutil.net\_if\_addrs().items():

        for snic in snics:

            if snic.family == sistema:

                yield (interface, snic.address, snic.netmask)

def get\_meus\_ips():

    return list(getNewIp(socket.AF\_INET))

def retorna\_codigo\_ping(hostname):

    """Usa o utilitario ping do sistema operacional para encontrar   o host. ('-c 5') indica, em sistemas linux, que deve mandar 5   pacotes. ('-W 3') indica, em sistemas linux, que deve esperar 3   milisegundos por uma resposta. Esta funcao retorna o codigo de   resposta do ping """

    plataforma = platform.system()

    args = []

    if plataforma == "Windows":

        args = ["ping", "-n", "1", "-l", "1", "-w", "100", hostname]

    else:

        args = ['ping', '-c', '1', '-W', '1', hostname]

    ret\_cod = subprocess.call(args, stdout=open(os.devnull, 'w'), stderr=open(os.devnull, 'w'))

    return ret\_cod

def get\_hosts\_rede(ip\_base):

    """Verifica todos os host com a base\_ip entre 1 e 255 retorna uma lista com todos os host que tiveram resposta 0 (ativo)"""

    host\_validos = []

    return\_codes = dict()

    for i in range(1, 255):

        return\_codes[ip\_base + '{0}'.format(i)] = retorna\_codigo\_ping(ip\_base + '{0}'.format(i))

        if i %20 ==0:

            print(".", end = "")

        if return\_codes[ip\_base + '{0}'.format(i)] == 0:

            host\_validos.append(ip\_base + '{0}'.format(i))

    return host\_validos

def detalhar\_host(host\_validos):

    """Obtendo nome do host"""

    nm = nmap.PortScanner()

    for host in host\_validos:

        try:

            nm.scan(host)

            host\_ = Host(host, nm[host].hostname())

            ## host\_ = Host(host, 'carlos-MS-7a38')

            for proto in nm[host].all\_protocols():

                print('----------')

                print('Protocolo : %s' % proto)

            lport = nm[host][proto].keys()

            for port in lport:

                port\_ = Porta(port, nm[host][proto][port]['state'])

                host\_.ports.append(port\_)

        except:

            pass

        portas = []

        for porta in host\_.ports:

            portas.append({ 'porta': porta.port, 'estado': porta.state})

        retorno = { 'ip' : host\_.ip, 'nome': host\_.name, 'portas': portas}

        variaveis['hosts\_detalhado'].append(retorno)

def get\_hosts():

    meus\_ips = variaveis['ips']

    if len(meus\_ips) > 0:

        meu\_ip\_principal = meus\_ips[0][1]

        # trata ip broadcast

        if meu\_ip\_principal == '127.0.0.1':

            meu\_ip\_principal = meus\_ips[1][1]

        print('Ip que será utilizado como base', meu\_ip\_principal)

        ip\_string = meu\_ip\_principal

        ip\_lista = ip\_string.split('.')

        base\_ip = ".".join(ip\_lista[0:3]) + '.'

        print("A busca será realizada na sub rede: ", base\_ip)

        hosts\_localizados = get\_hosts\_rede(base\_ip) #['192.168.0.12', '192.168.0.13', '192.168.0.14']

        print('Verificar nomes dos hosts', hosts\_localizados, '\r')

        detalhar\_host(hosts\_localizados)

def get\_trafego\_host():

    io\_status = psutil.net\_io\_counters(pernic=True)

    hosts = variaveis['ips']

    trafego\_interface = []

    if len(hosts) > 0:

        for host in hosts:

            trafego = io\_status[host[0]]

            enviado = trafego[0]

            recebido = trafego[1]

            pct\_enviado = trafego[2]

            pct\_recebido = trafego[3]

            trafego\_aux = Trafego(host[0], enviado, recebido, pct\_enviado, pct\_recebido)

            trafego\_interface.append(trafego\_aux.to\_map())

        variaveis['trafego'].append(trafego\_interface)

def get\_info\_memoria():

    memoria = psutil.virtual\_memory()

    capacidade = round(memoria.total/(1024\*\*3), 2)

    disponivel = round(memoria.available/(1024\*\*3), 2)

    memoria\_aux = Memoria(memoria, capacidade, disponivel)

    variaveis['memoria'].append(memoria\_aux.get\_map())

def get\_processo():

    pids = psutil.pids()

    processos\_coletados = []

    for pid in pids:

        try:

            nome = psutil.Process(pid).name()

            percent\_uso = psutil.Process(pid).memory\_percent()

            memoria\_usada = psutil.Process(pid).memory\_info().rss / 1024/1024

            threads\_usadas = psutil.Process(pid).num\_threads()

            tempo\_usuario = str(psutil.Process(pid).cpu\_times().user) + ' s'

            data\_criacao = time.ctime(psutil.Process(pid).create\_time())

            processo\_aux = Processo(pid, nome, percent\_uso, memoria\_usada, threads\_usadas, tempo\_usuario, data\_criacao)

            processos\_coletados.append(processo\_aux.to\_map())

        except:

            print('>>> Erro ao obter informações sobre o processo de pid: ', pid)

    return processos\_coletados

def get\_processos\_pagina(pagina, processos):

    total\_de\_processo = len(processos)

    total\_paginas = math.ceil(total\_de\_processo / variaveis['total\_elementos\_por\_pagina'])

    limite = int(pagina) \* variaveis['total\_elementos\_por\_pagina']

    inicio = limite - variaveis['total\_elementos\_por\_pagina']

    paginado = processos[inicio:limite]

    return {

        'elementos': paginado,

        'pagina\_atual': pagina,

        'total\_paginas': total\_paginas,

        'total\_elementos\_por\_pg': variaveis['total\_elementos\_por\_pagina'],

        'total\_processos': total\_de\_processo,

    }

def get\_arquivos\_paginado(pagina, arquivos):

    total\_de\_arquivos = len(arquivos)

    total\_paginas = math.ceil(total\_de\_arquivos / variaveis['total\_elementos\_por\_pagina'])

    limite = int(pagina) \* variaveis['total\_elementos\_por\_pagina']

    inicio = limite - variaveis['total\_elementos\_por\_pagina']

    paginado = arquivos[inicio : limite]

    return {

        'elementos': paginado,

        'pagina\_atual': pagina,

        'total\_paginas': total\_paginas,

        'total\_elementos\_por\_pg': variaveis['total\_elementos\_por\_pagina'],

        'total\_arquivos': total\_de\_arquivos,

    }

# fim funcoes

# inicia thread

thread\_disco = ThreadDisco(1, 'Thread-Disco', 1)

thread\_disco.start()

thread\_cpu = ThreadCpu(1, 'Thread-CPU', 1)

thread\_cpu.start()

thread\_sched = ThreadSched(1, 'Thread-sched', 1)

thread\_sched.start()

thread\_rede = ThreadIps(1, 'Thread-rede', 1)

thread\_rede.start()

thread\_scan\_rede = ThreadRede(1, 'Thread-scan-rede', 1)

thread\_scan\_rede.start()

thread\_scan\_trafego\_rede = ThreadTrafegoRede(1, 'Thread-trafego-rede', 1)

thread\_scan\_trafego\_rede.start()

thread\_memoria = ThreadMemoria(1, 'Thread-memoria', 1)

thread\_memoria.start()

thread\_processo = ThreadProcesso(1, 'Thread-processo', 1)

thread\_processo.start()

# inicio infra servidor

socket\_servidor = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

host = socket.gethostname()

porta = 9999

socket\_servidor.bind((host, porta))

socket\_servidor.listen()

print("Servidor de nome:", host, " - Aguardando conexão na porta:", porta)

(socket\_cliente,addr) = socket\_servidor.accept()

print("Conectado a:", str(addr))

# fim infra servidor

while True:

    response = 'NoNe'

    try:

        msg = socket\_cliente.recv(11)

    except:

        print('>>>> Coexão perdida')

        print("Servidor de nome:", host, " - Aguardando conexão na porta:", porta)

        (socket\_cliente,addr) = socket\_servidor.accept()

    decode = ''

    pagina = 1

    try:

        decode\_aux = msg.decode('ascii')

        decode\_aux = decode\_aux.split('/')

        decode = decode\_aux[0]

        pagina = decode\_aux[1]

    except:

        decode = msg.decode('ascii')

    if decode == 'fim':

        break

    elif decode == 'disco':

        disco\_aux = variaveis['disco'][len(variaveis['disco']) - 1]

        response = disco\_aux.to\_map()

    elif decode == 'cpu':

        cpu\_aux = variaveis['cpu'][len(variaveis['cpu']) - 1]

        response = cpu\_aux.to\_map()

    elif decode == 'arquivos':

        response = []

        sched\_aux = variaveis['sched'][len(variaveis['sched']) - 1]

        arquivos\_aux = variaveis['arquivos'][len(variaveis['arquivos']) - 1]

        result = get\_arquivos\_paginado(pagina, arquivos\_aux)

        diretorio = os.getcwd()

        response.append(diretorio)

        response.append(sched\_aux)

        response.append(result)

    elif decode == 'ips':

        response = variaveis['ips']

    elif decode == 'rede':

        if len(variaveis['hosts\_detalhado']) > 0:

            response = variaveis['hosts\_detalhado']

    elif decode == 'trafego':

        if len(variaveis['trafego']) > 0:

            response = variaveis['trafego']

    elif decode == 'memoria':

        if len(variaveis['memoria']) > 0:

            response = variaveis['memoria'][len(variaveis['memoria']) - 1]

    elif decode == 'processo':

        processos = variaveis['processo']

        response = get\_processos\_pagina(pagina, processos)

    elif decode == 'resumo':

        total\_processo = len(variaveis['processo'])

        memoria = variaveis['memoria'][len(variaveis['memoria']) - 1]

        cpu\_aux = variaveis['cpu'][len(variaveis['cpu']) - 1]

        cpu\_aux = cpu\_aux.to\_map()

        disco\_aux = variaveis['disco'][len(variaveis['disco']) - 1]

        disco\_aux = disco\_aux.to\_map()

        ips = variaveis['ips']

        resumo = Resumo(total\_processo, memoria['capacidade'], memoria['disponivel'], cpu\_aux, disco\_aux)

        resumo.ips = ips

        response = resumo.to\_map()

    bytes\_resp = pickle.dumps(response)

    socket\_cliente.send(bytes\_resp)

# Fecha socket do servidor e cliente

socket\_cliente.close()

socket\_servidor.close()

# Telas

## Evidência v1

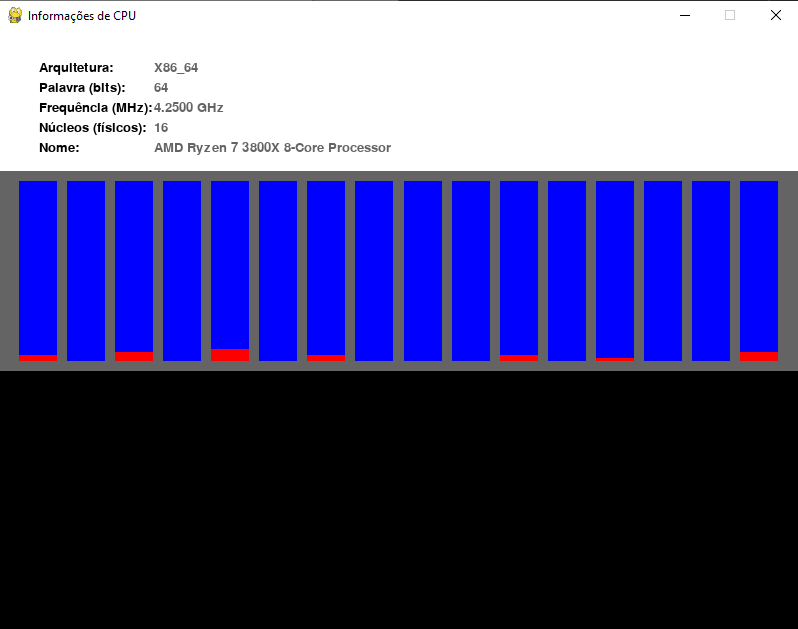


Figure Processador

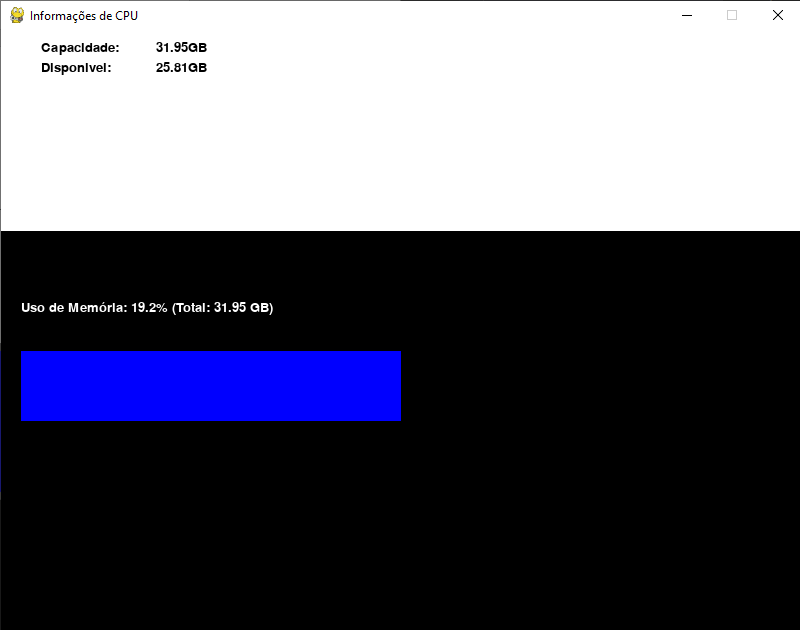


Figure Memória



Figure Disco



Figure Rede

## Evidências v2

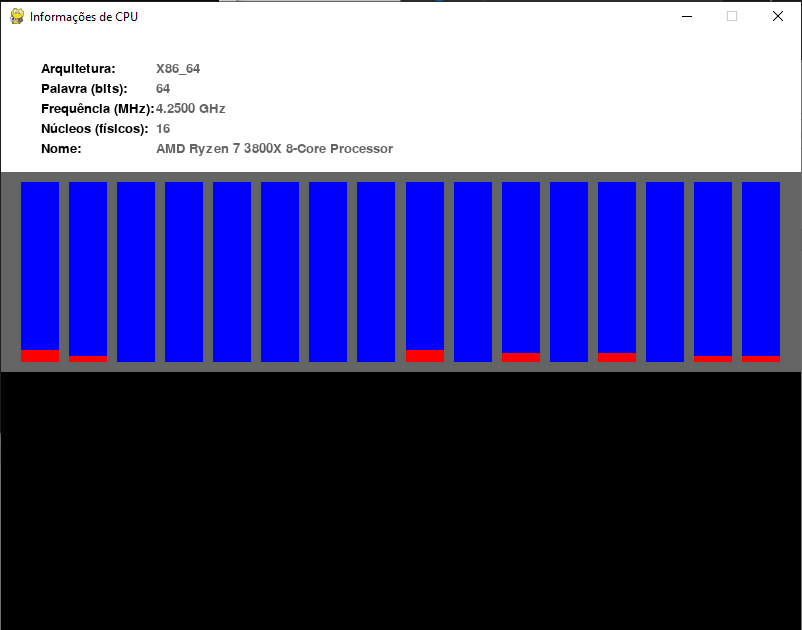


Figure Processador



Figure Memória



Figure Disco

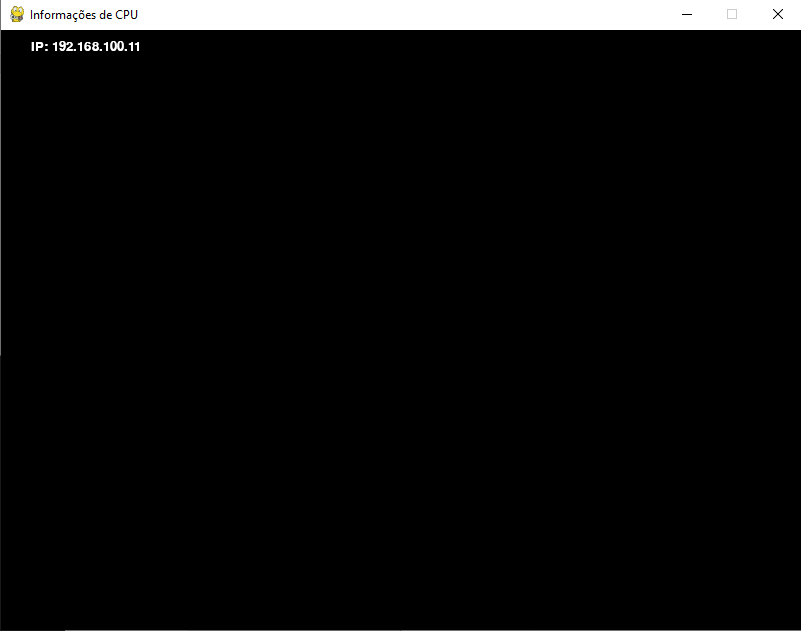


Figure Rede

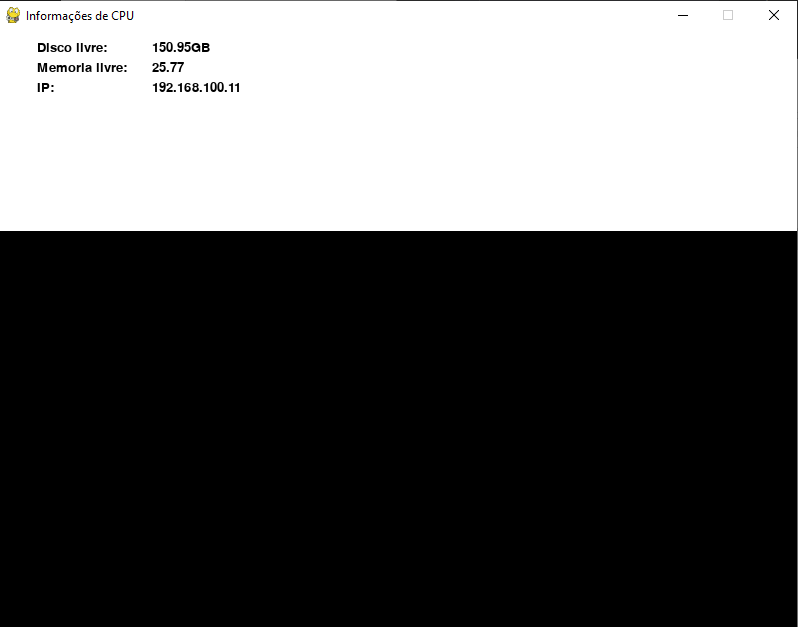


Figure Resumo

## Evidências v3

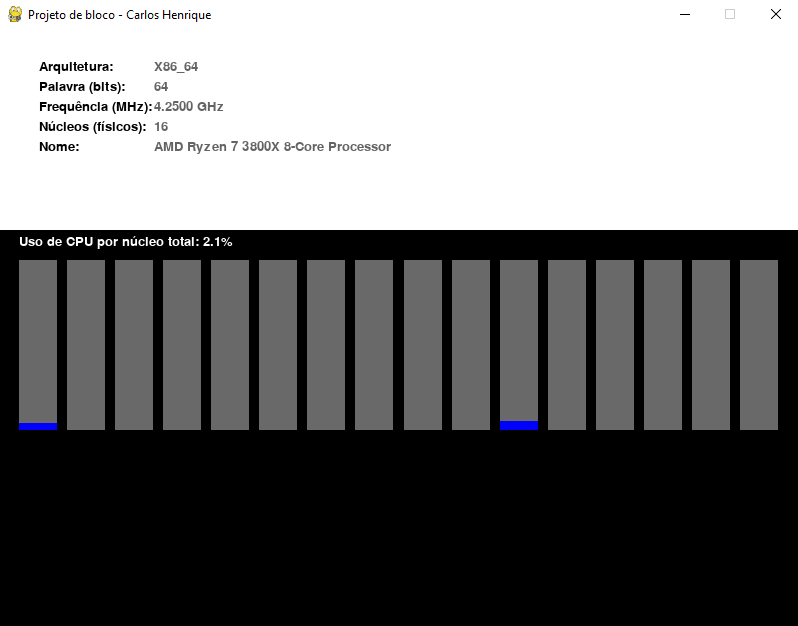


Figure Processador

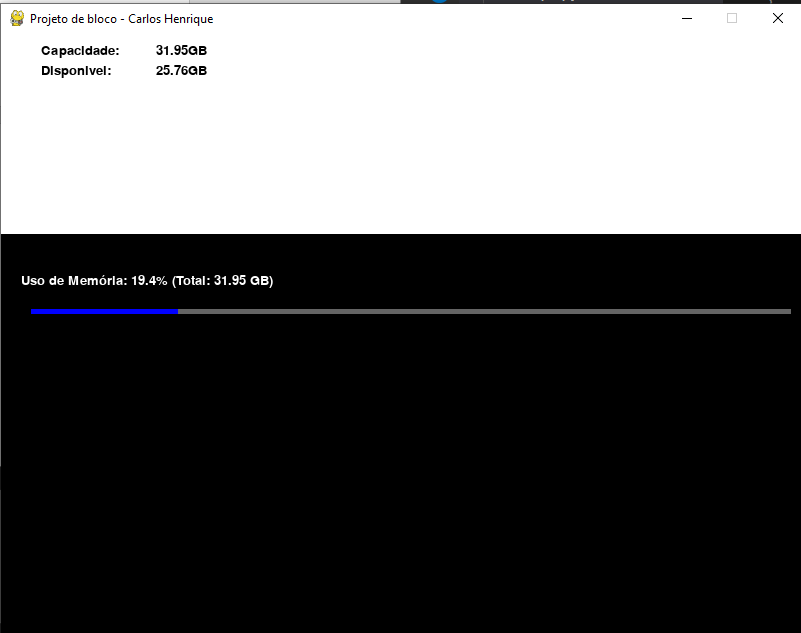


Figure Memória

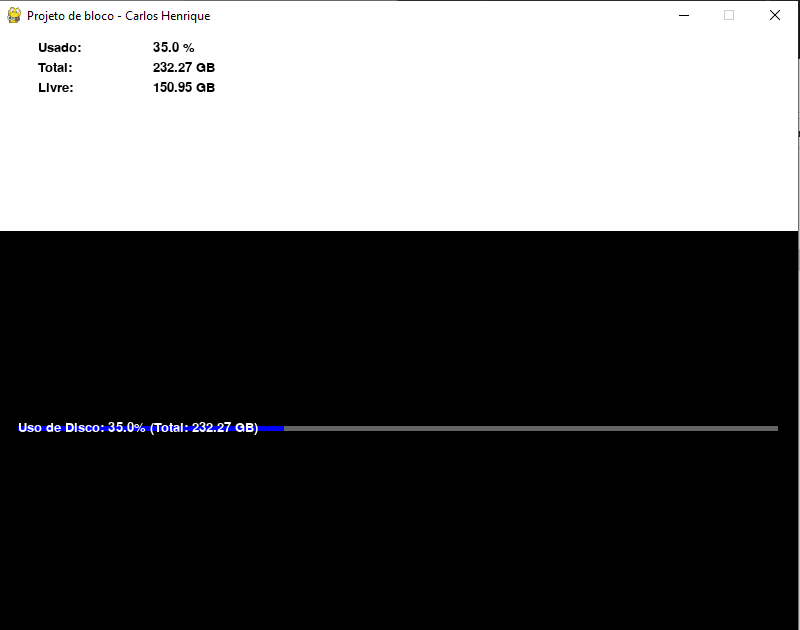


Figure Disco

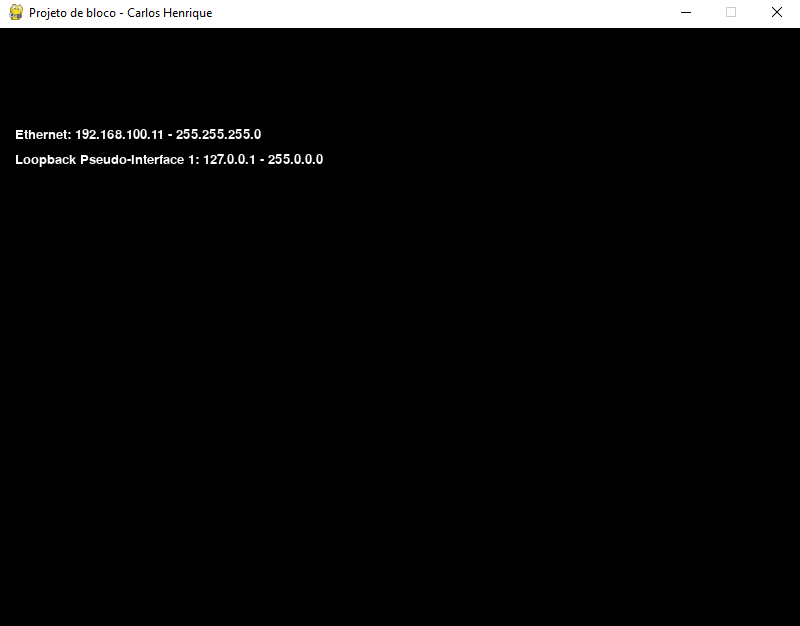


Figure Rede

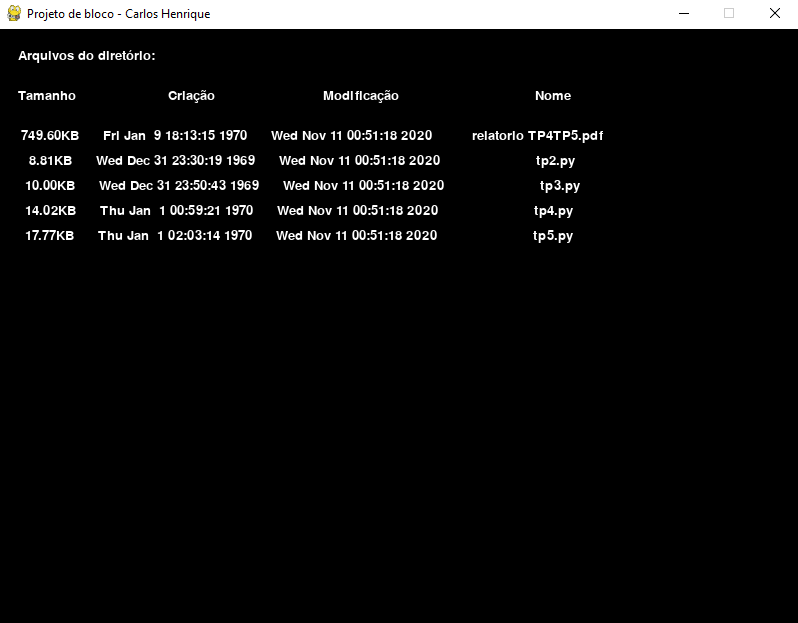


Figure Diretórios

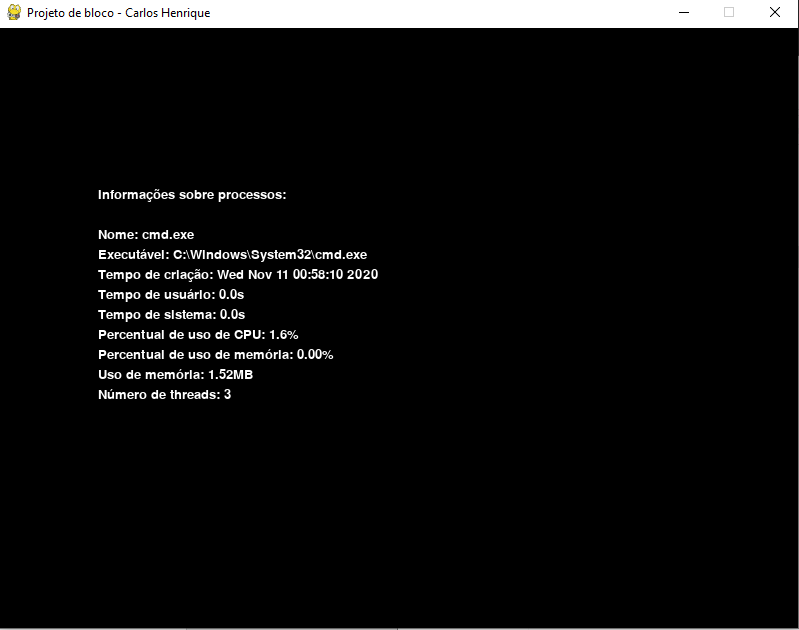


Figure Processo

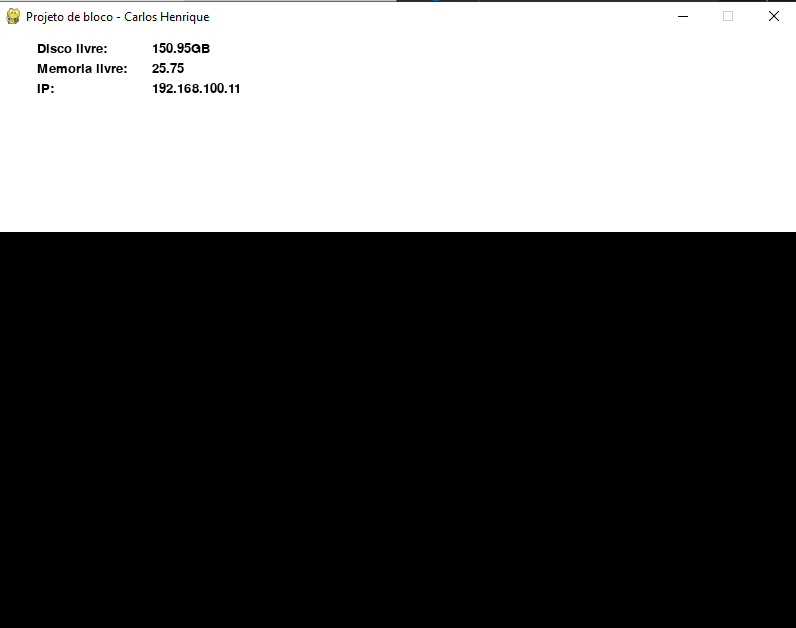


Figure Resumo

## Evidências v6

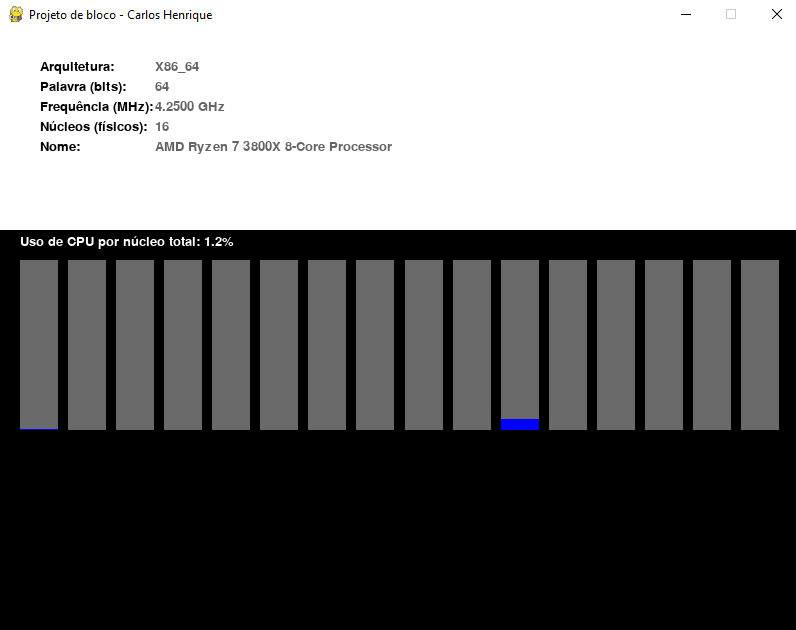


Figure Processador

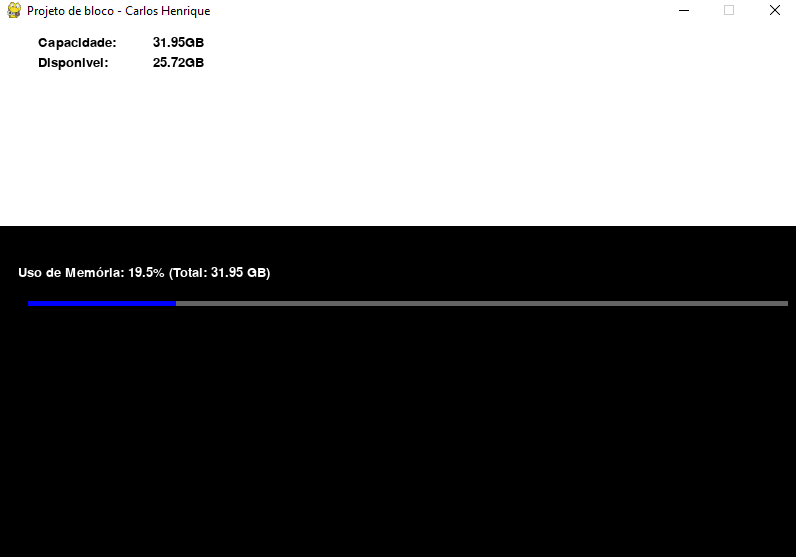


Figure Memória

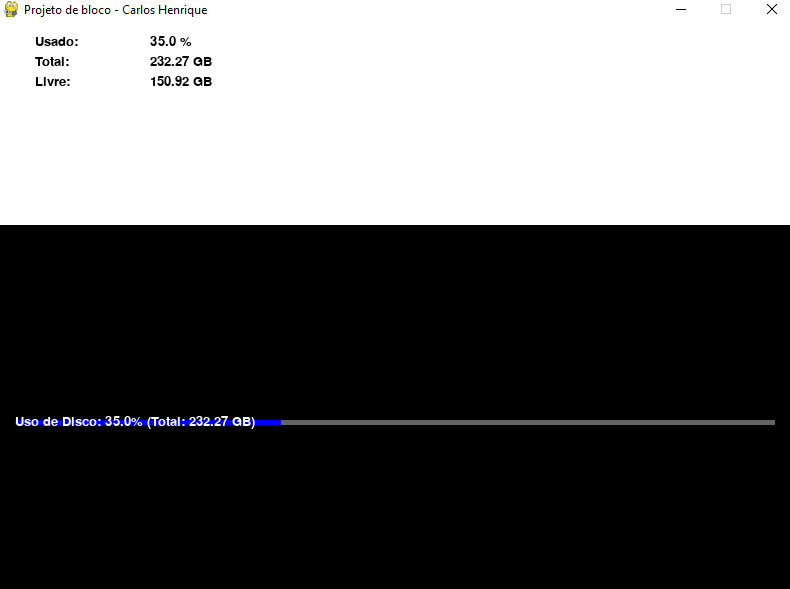


Figure Disco

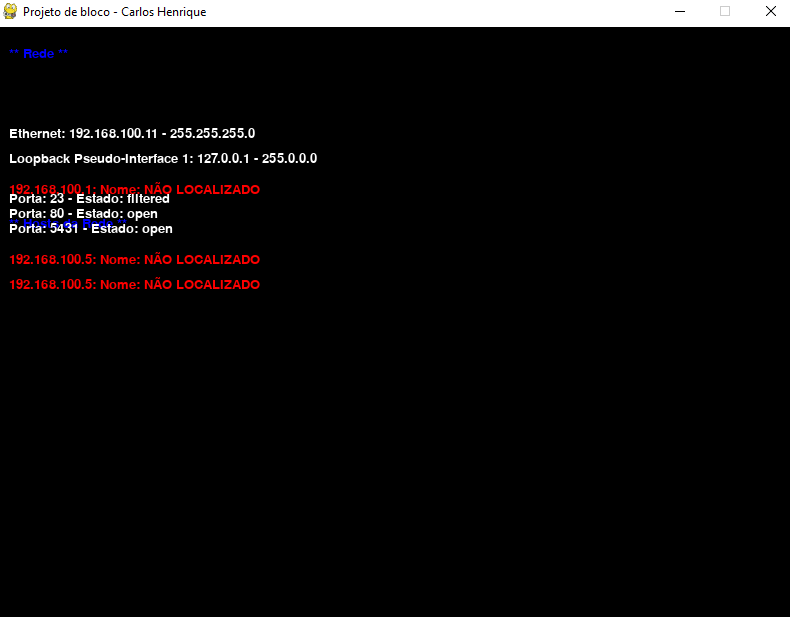


Figure Rede

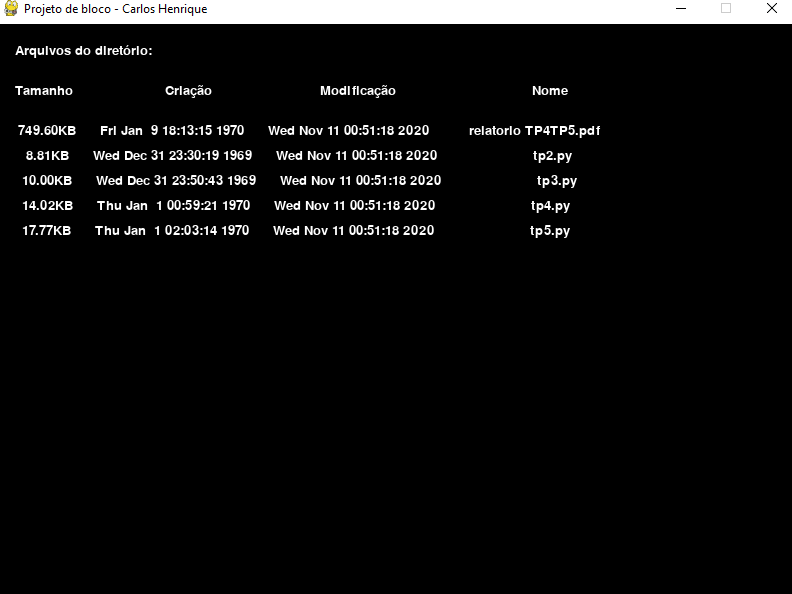


Figure Arquivos

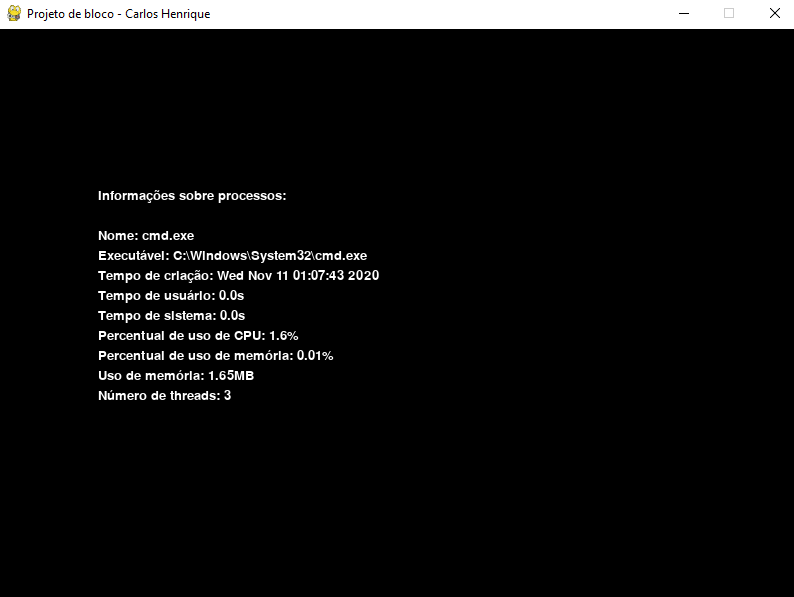


Figure Processo

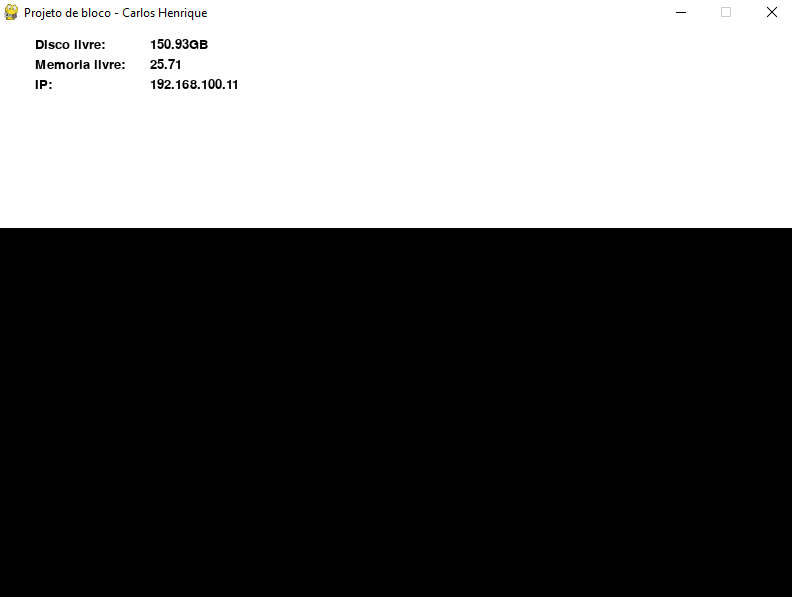


Figure Rede

## Evidências v5

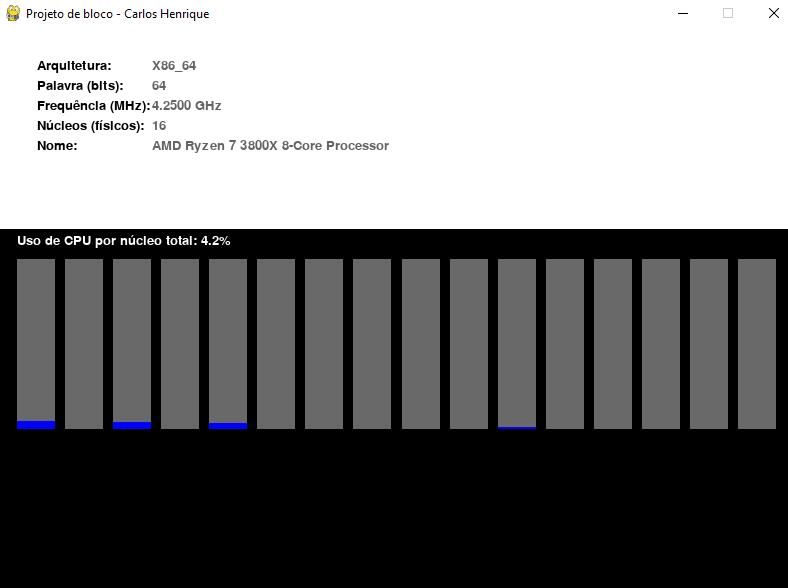


Figure CPU

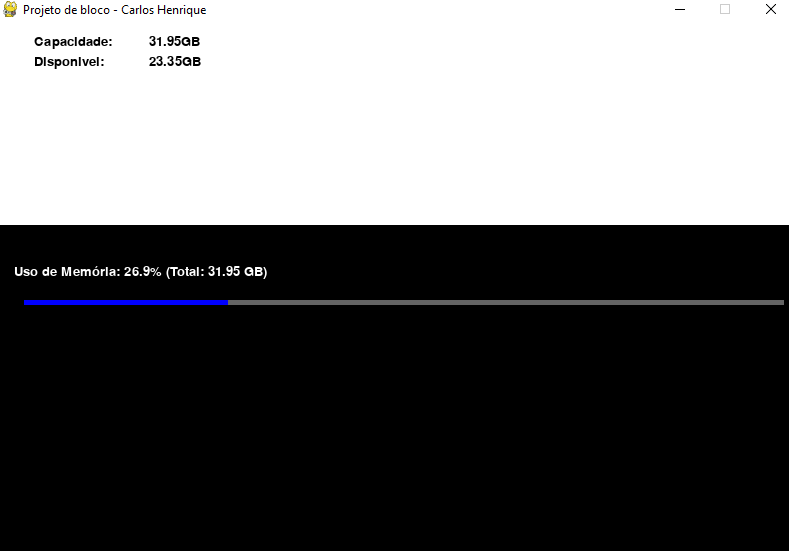


Figure Memória

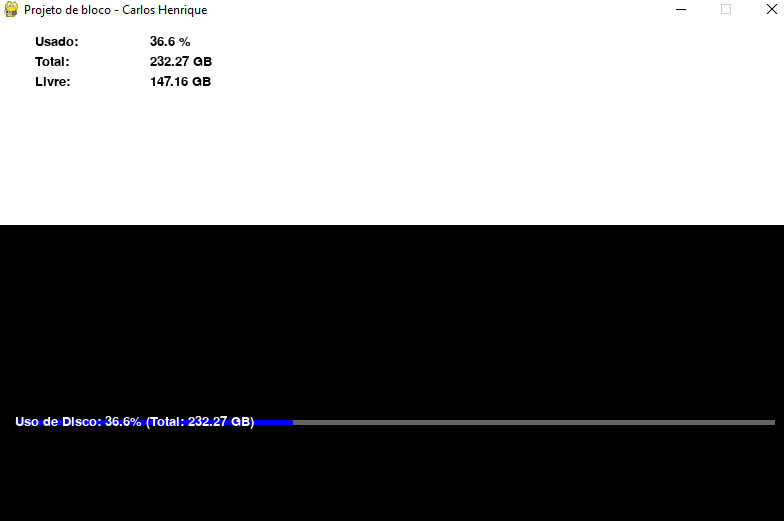


Figure Disco

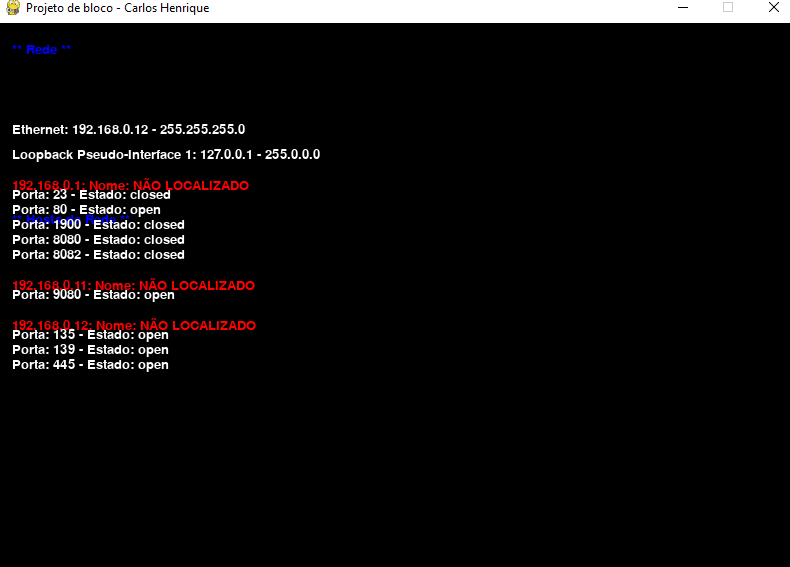


Figure Rede

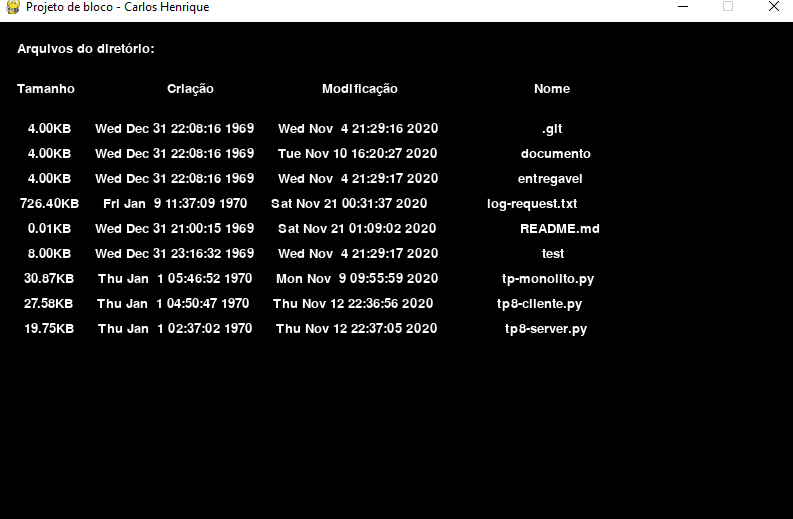


Figure Diretório

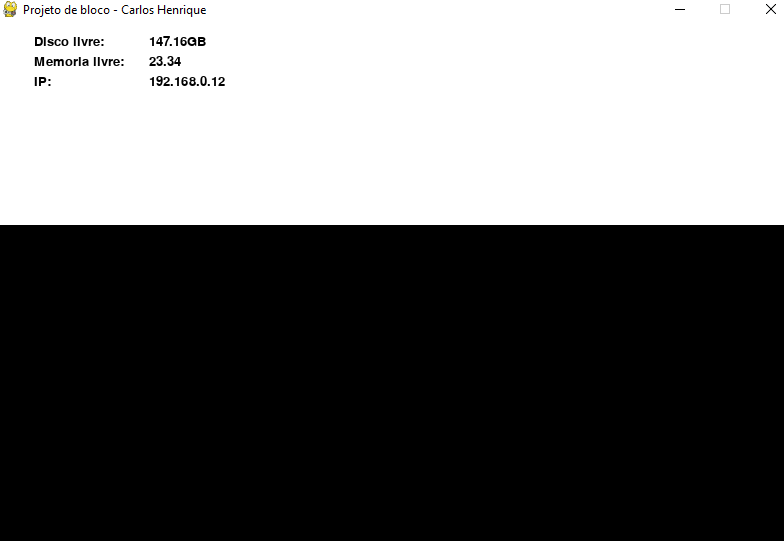


Figure Resumo

## Evidências v6:

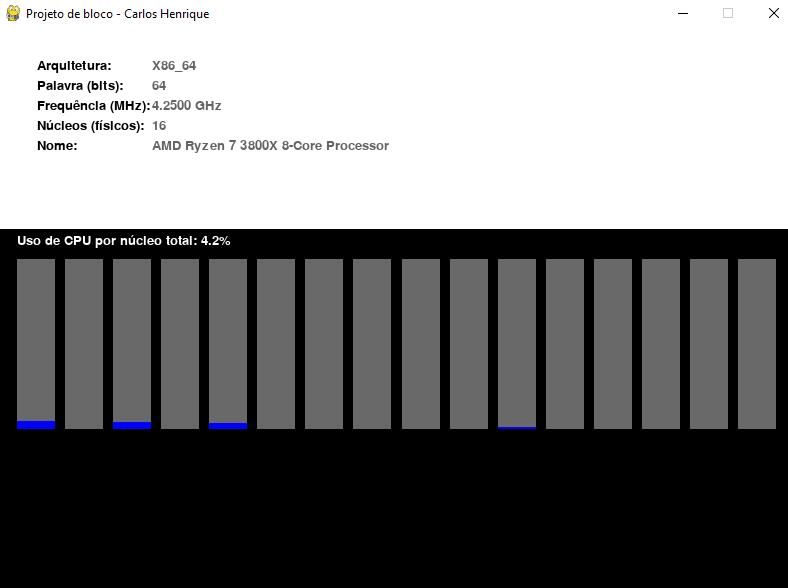


Figure Processador

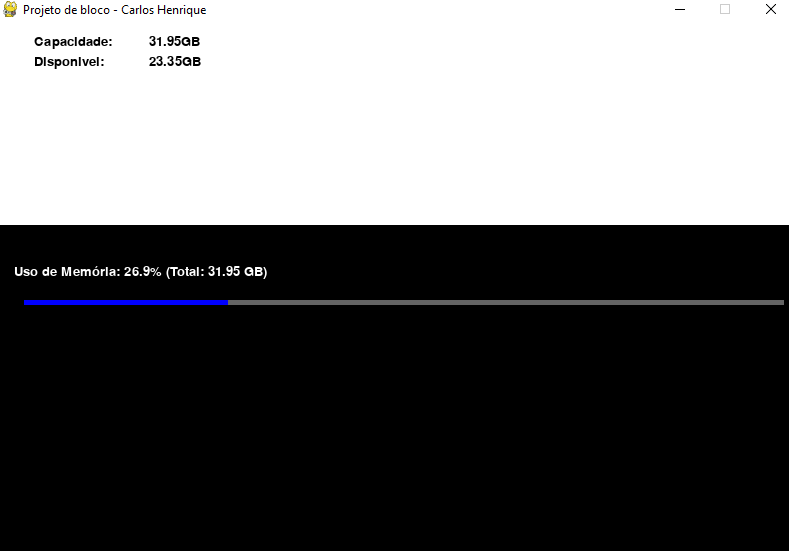


Figure Memória

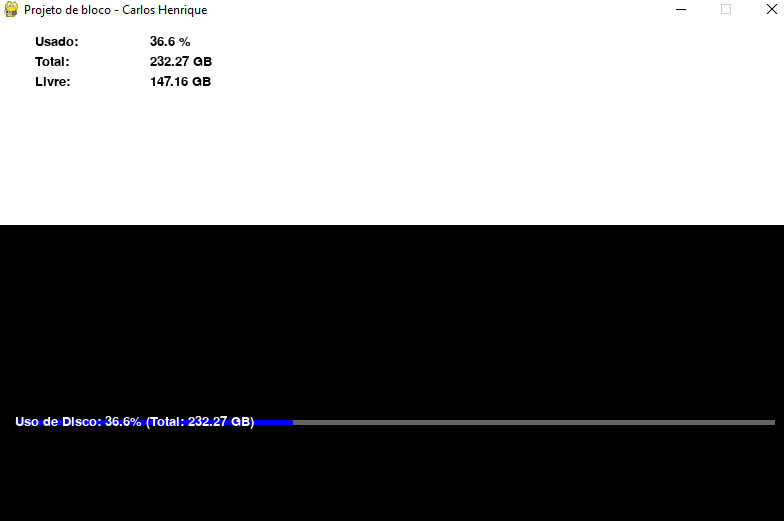


Figure Disco

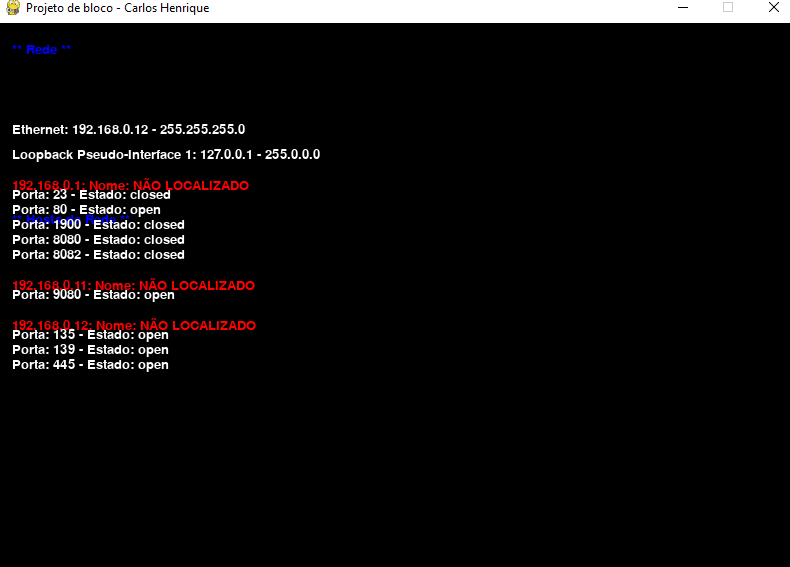


Figure Rede

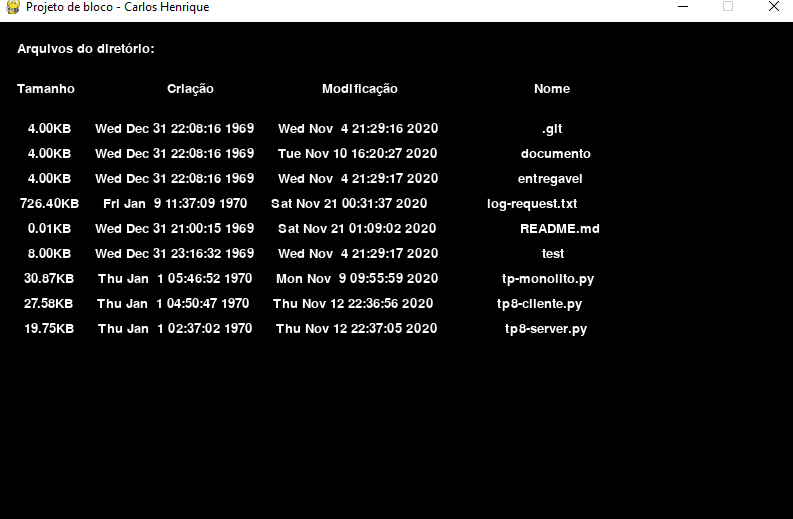


Figure Diretório

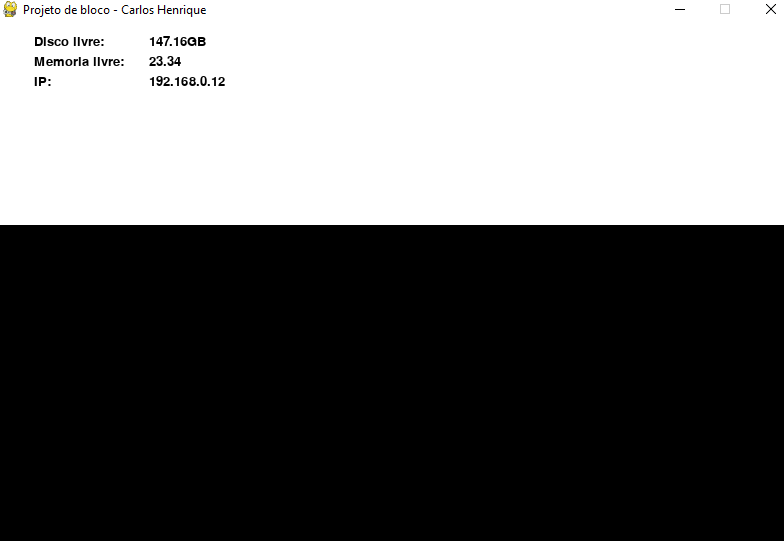


Figure Resumo

## Evidência v7:

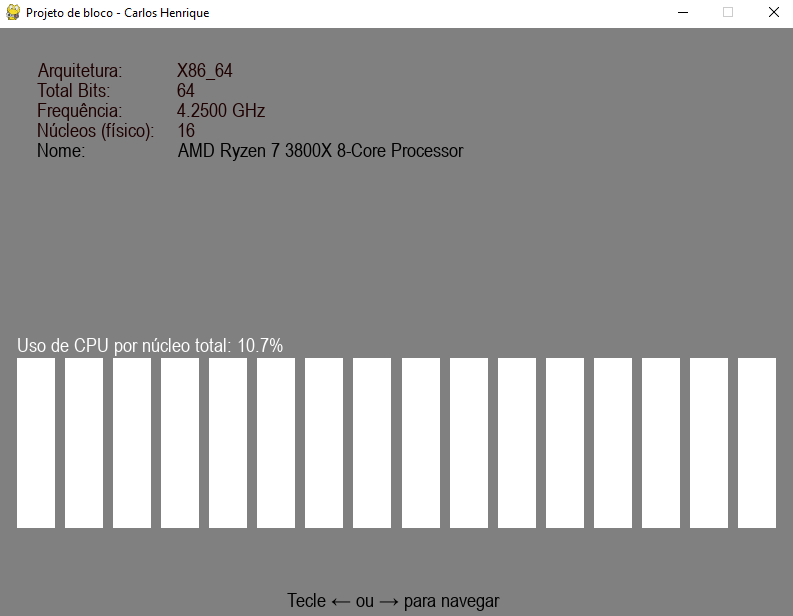


Figure Processador

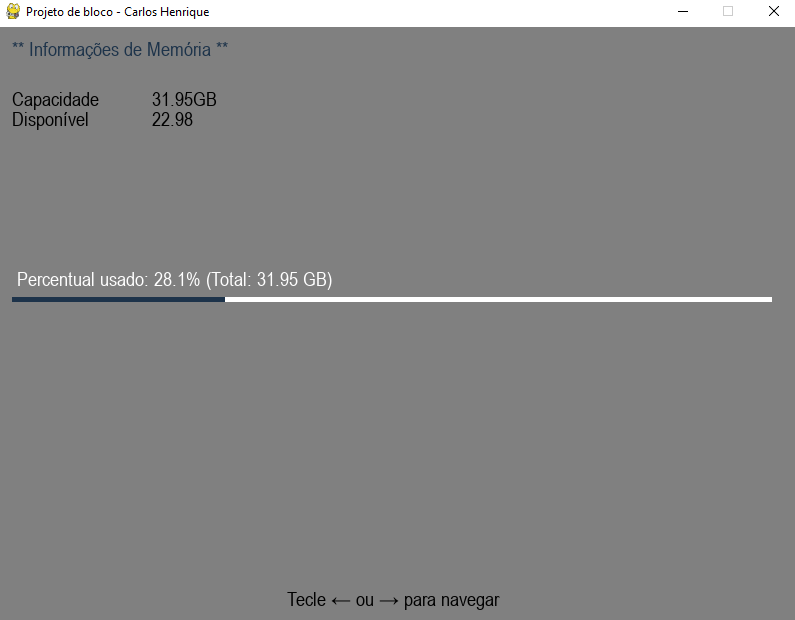


Figure Memória

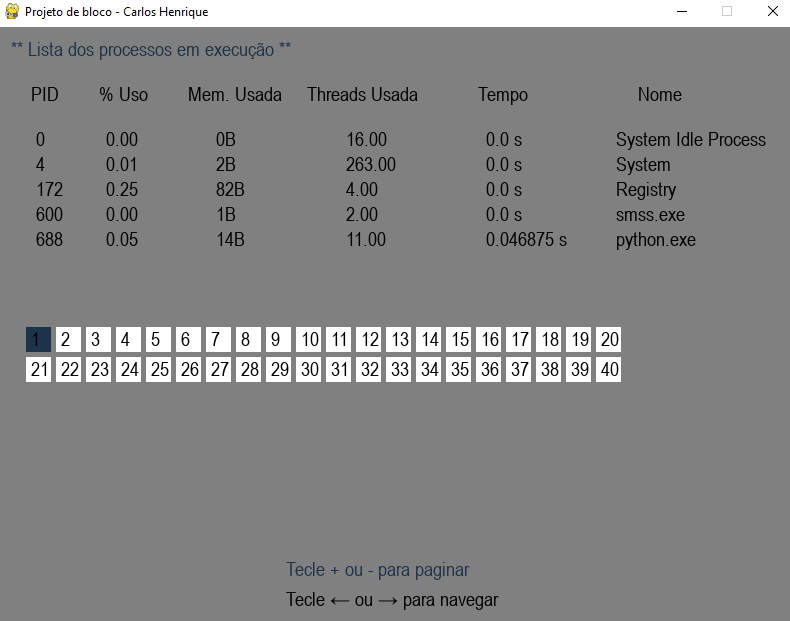


Figure Processos

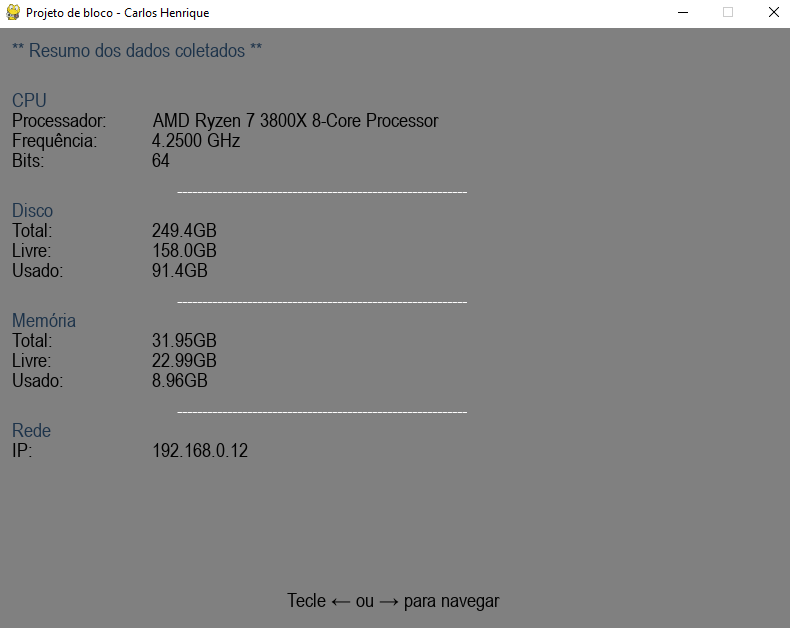


Figure Resumo

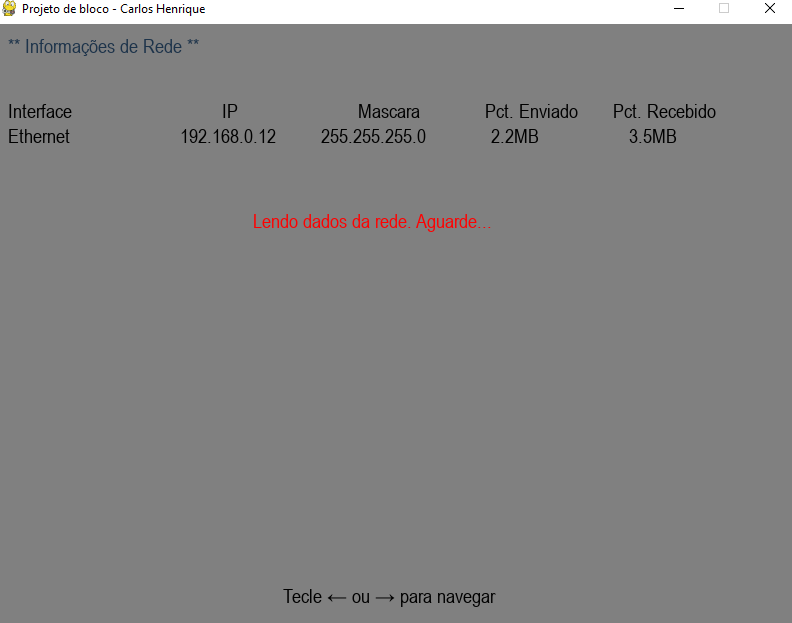


Figure Rede em análise

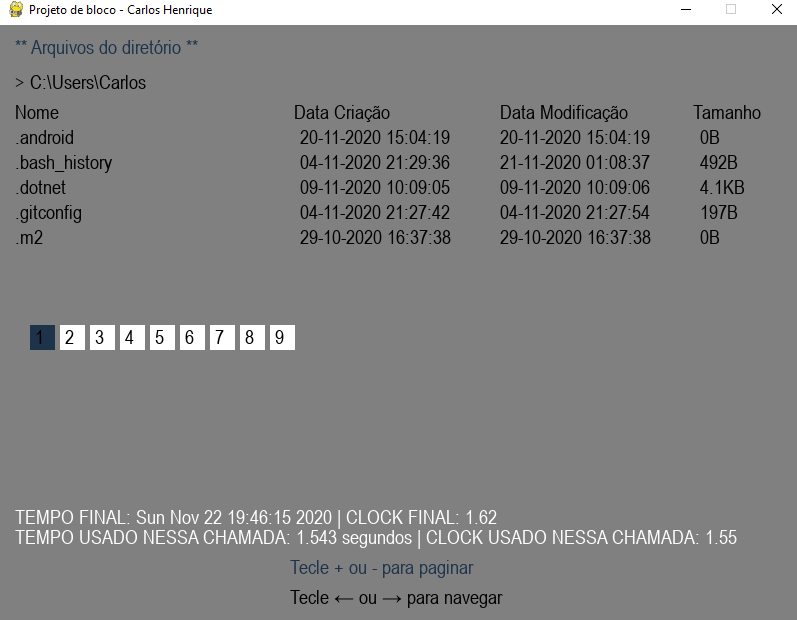


Figure Arquivos

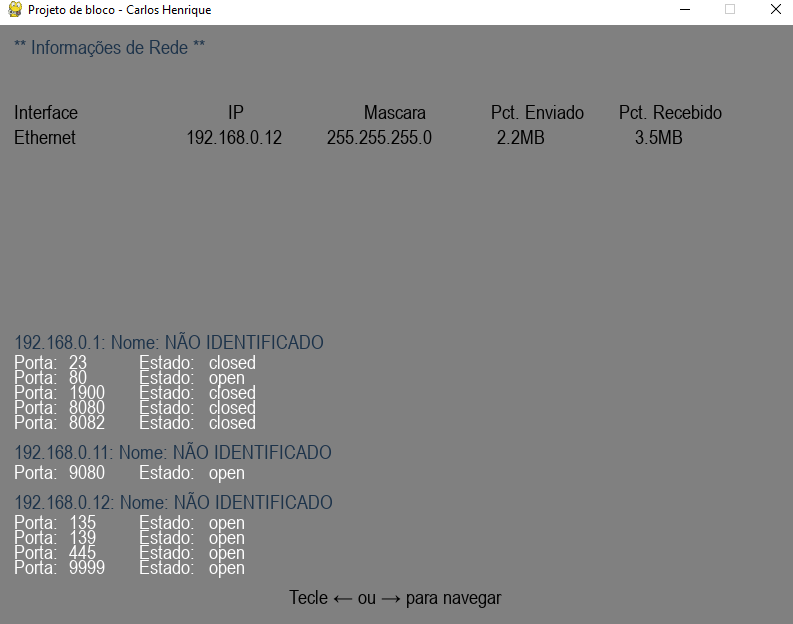


Figure Rede analisada

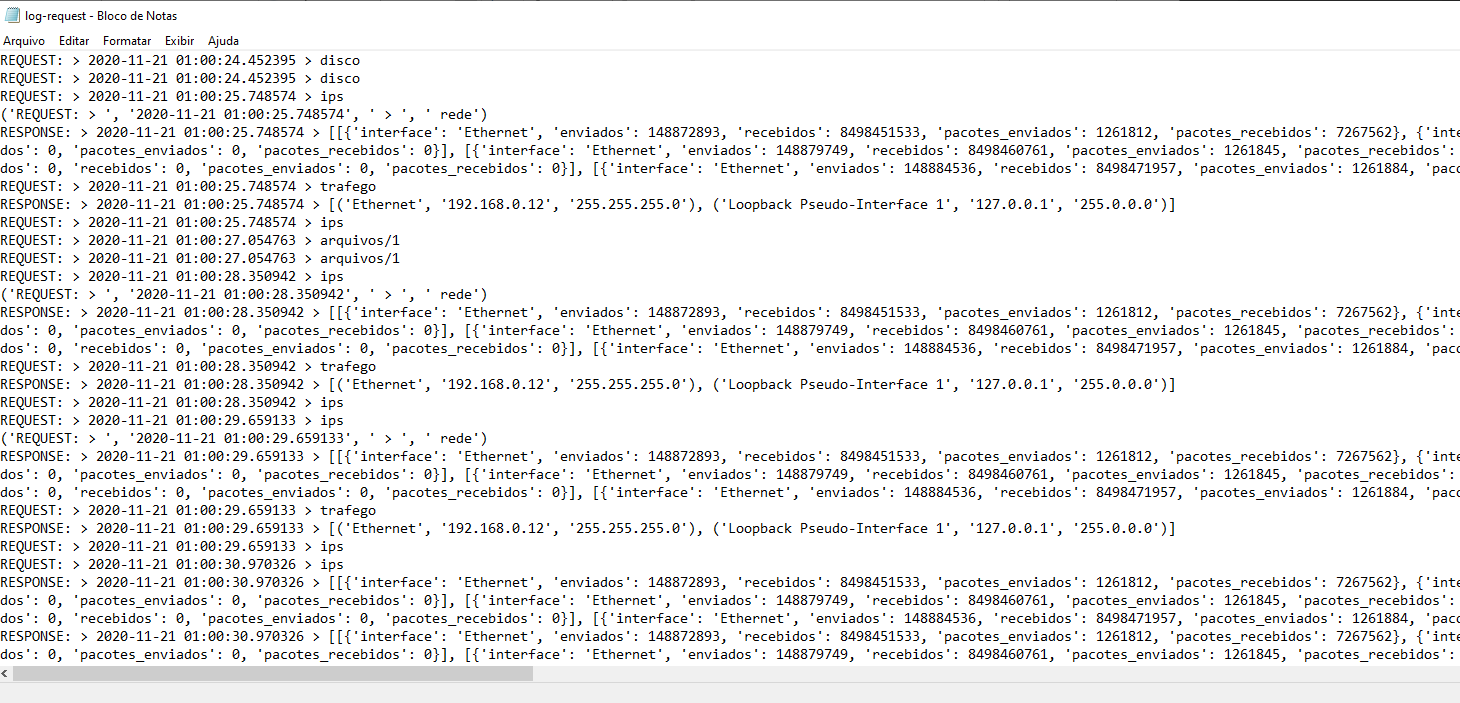


Figure Log