

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
Objetivo	6
Bibliotecas utilizadas	6
Desenvolvimento.....	6
Problemas observados:	7
Soluções de problemas anteriores:.....	7
Capítulo 2	8
Objetivo	8
Bibliotecas utilizedas.....	8
Desenvolvimento.....	8
Problemas observados	10
Soluções de problemas anteriores.....	10
Capítulo 3	11
Objetivo	11
Bibliotecas utilizadas	11
Desenvolvimento.....	11
Problemas observados:	12
Soluções de problemas anteriores:.....	12
Correção proporção do tamanho do gráfico de Disco	12
Correção proporção do tamanho do gráfico de Memória.....	12
Correção carrossel	12
Capítulo 4	14
Objetivo	14
Bibliotecas utilizadas	14
Desenvolvimento.....	14
Problemas observados:	15
Soluções de problemas anteriores:.....	15
Capítulo 5	16
Objetivo	16
Bibliotecas utilizadas	16
Desenvolvimento.....	16

Infraestrutura	16
Problemas observados	16
Soluções de problemas anteriores.....	17
Capítulo 6	18
Objetivo	18
Bibliotecas.....	18
Desenvolvimento.....	18
Problemas observados:	19
Soluções de problemas anteriores:.....	19
Capítulo 7	20
Objetivo.....	20
Bibliotecas utilizadas	20
Cliente.....	20
Servidor	20
Desenvolvimento.....	20
Problemas observados	22
Soluções de problemas anteriores.....	22
Anexos	23
Código fonte v1	24
Código fonte v2	30
Código fonte v3	37
Código fonte v4	46
Código fonte v5	57
Código fonte v6	69
Código fonte v7	81
Cliente:.....	81
servidor:.....	99
Telas	114
Evidência v1.....	115
Evidências v2	117
Evidências v3	120
Evidências v6	124
Evidências v5	128
Evidências v6:.....	131
Evidência v7:.....	134

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 **platform** e **psutil** instalado. O pacote **platform** 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.

A fim de obter os dados da memória, foi utilizado a função `virtual_memory()`, que retorna dados estatísticos da memória do computador. Essa função retorna um objeto 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 utilizadas

- 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 “**cputinfo**” e a função “**get_cpu_info()**”. Onde o retorno obtido foi:

```
('python_version': '3.9.0.final.0 (64 bit)', 'cputinfo_version': [7, 0, 0], 'cputinfo_version_string': '7.0.0', 'arch': 'X86_64', 'bits': 64, 'count': 16, 'arch_string_raw': 'AMD64', 'vendor_id_raw': 'AuthenticAMD', 'brand_raw': 'AMD Ryzen 7 3800X 8-Core Processor', 'hz_actual_friendly': '4.2500 GHz', 'hz_actual': [4250000000, 0], 'l2_cache_size': 4194304, 'model': 113, 'family': 23, 'l3_cache_size': 33554432, 'hz_advertised_friendly': '4.2500 GHz', 'hz_advertised': [4250000000, 0], 'flags': ['3dnow', '3dnowprefetch', 'abm', 'adx', 'aes', 'apic', 'avx', 'avx2', 'bm1', 'bm12', 'clflush', 'clflushopt', 'clwb', 'cmov', 'cmp_legacy', 'cr8_legacy', 'cx16', 'cx8', 'dbx', 'de', 'dts', 'extapic', 'f16c', 'fma', 'fpu', 'fxsr', 'ht', 'ia64', 'ibs', 'lahf_lm', 'lm', 'mca', 'mce', 'misalignsse', 'mmx', 'monitor', 'movbe', 'msr', 'mtsr', 'osw', 'osxsave', 'pae', 'pat', 'pc128i', 'pcimuldq', 'perfctr_core', 'perfctr_nb', 'pge', 'pn1', 'popcnt', 'pqr', 'pqn', 'pse', 'pse36', 'rdpid', 'rdmnid', 'rdseed', 'sep', 'sepamd', 'serial', 'sha', 'skinit', 'snap', 'smep', 'ss', 'sse', 'sse2', 'sse4_1', 'sse4_2', 'sse4a', 'ssse3', 'svn', 'tce', 'tm', 'topoext', 'tsc', 'umip', 'vme', 'wdt', 'xsav'], 'l2_cache_line_size': 512, 'l2_cache_associativity': 6}
```

Figura 1 Resposta CpuInfo

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

- 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:

```
sdiskusage(total=249403043840, used=87985115136, free=161417928704, percent=35.3)
```

Para se obter as informações exigidas nesse capítulo, 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:

disco.used/_{1024**3}

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:

```
{'Ethernet': [snicaddr(family=<AddressFamily.AF_LINK: -1>, address='2C-F0-5D-08-DB-65', netmask=None, broadcast=None, ptp=None), snicaddr(family=<AddressFamily.AF_INET: 2>, address='192.168.100.11', net mask='255.255.255.0', broadcast=None, ptp=None), snicaddr(family=<AddressFamily.AF_INET6: 23>, address='fe80::3d66:f0ae:9e7:dd6d', netmask=None, broadcast=None, ptp=None)], 'Loopback Pseudo-Interface 1': [snicaddr(family=<AddressFamily.AF_INET: 2>, address='127.0.0.1', netmask='255.0.0.0', broadcast=None, ptp=None), snicaddr(family=<AddressFamily.AF_INET6: 23>, address='::1', netmask=None, broadcast=None, ptp=None)]}
```

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:

```
['.git', 'documento', 'entregavel', 't8.py', 'test']
```

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. PID 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:

```
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))
```

Figura 2 Código para apresentar o consumo do disco

Correção proporção do tamanho do gráfico de Memória

```
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))
```

Figura 3 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:

```
#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
```

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_scheduler_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:

```
TEMPO FINAL: Thu Nov 12 17:17:09 2020 | CLOCK FINAL: 0.38
TEMPO USADO NESSA CHAMADA: 0.002 segundos | CLOCK USADO NESSA CHAMADA: 0.00
('TEMPO FINAL: Thu Nov 12 17:17:09 2020 | CLOCK FINAL: 0.38', 'TEMPO USADO NESSA CHAMADA: 0.002 segundos | CLOCK USADO NESSA CHAMADA: 0.00')
```

Problemas observados:

- No capítulo 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:

```
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))
```

Resultado obtido:

```
[('Ethernet', '192.168.0.12', '255.255.255.0'), ('Loopback Pseudo-Interface 1', '127.0.0.1', '255.0.0.0')]
```

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 parâmetro 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 capítulo, 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 tráfego 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** e 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 requisição de 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

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:", 1
0)
    mostrar_texto_disco(superficie_info_disco, "total_disco", "Total:", 3
0)
    mostrar_texto_disco(superficie_info_disco, "disco_livre", "Livre:", 5
0)
    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()
#####

#####
```

```

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:", 1
0)
    mostrar_texto_disco(superficie_info_disco, "total_disco", "Total:", 3
0)
    mostrar_texto_disco(superficie_info_disco, "disco_livre", "Livre:", 5
0)

    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 li
vre:", 10)
    mostrar_texto_resumo(superficie_info_resumo, "memoria_livre", "Memori
a 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", "Capacid
ade:", 10)
    mostrar_texto_memoria(superficie_info_memoria, "disponivel", "Disponi
vel:", 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:", 1
0)
    mostrar_texto_disco(superficie_info_disco, "total_disco", "Total:", 3
0)
    mostrar_texto_disco(superficie_info_disco, "disco_livre", "Livre:", 5
0)
    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", "Memória 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:", 1
0)
    mostrar_texto_disco(superficie_info_disco, "total_disco", "Total:", 3
0)
    mostrar_texto_disco(superficie_info_disco, "disco_livre", "Livre:", 5
0)
    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", "Memória 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 um a resposta. Esta função retorna o código 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:", 1
0)
    mostrar_texto_disco(superficie_info_disco, "total_disco", "Total:", 3
0)
    mostrar_texto_disco(superficie_info_disco, "disco_livre", "Livre:", 5
0)

    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", "Memória 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", "Capacidadade:", 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(percent
agem, 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_modificac
ao + 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 ho
st. ('-c 5') indica, em sistemas linux, que deve mandar 5 pacotes. ('-
W 3') indica, em sistemas linux, que deve esperar 3 milisegundos por um
a 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 lis
ta 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:", 1
0)
    mostrar_texto_disco(superficie_info_disco, "total_disco", "Total:", 3
0)
    mostrar_texto_disco(superficie_info_disco, "disco_livre", "Livre:", 5
0)
    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(s1, "disco_livre", "Disco livre:", 10)
    mostrar_texto_resumo(s1, "memoria_livre", "Memória livre:", 30)
    mostrar_texto_resumo(s1, "ip_rede", "IP:", 50)
    tela.blit(s1, (0, 0))

def mostrar_texto_resumo(s1, nome, nome2, 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", "Capacidad", 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(percent
agem, 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_modificac
ao + 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 um a 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 informaçoes 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_frequencia = font.render(cpu['frequencia'], True, (30, 0, 0))
superficie_info_cpu.blit(valor_frequencia, (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", 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, v
ariaveis['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, variavei
s['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_cria-
cao']).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, va-
riaveis['preto'])
        tela.blit(data_modificacao_formatado, (500, espacos))

        tamanho_arquivo_formatado = font.render(tamanho_arquivo, 1, varia-
veis['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, var-
iaveis['azul'])
    tela.blit(titulo, (15, 10))

    titulo = font.render("      PID          % Uso          Mem. Usada      Thre-
ads Usada          Tempo          Nome" , 1, variaveis['pre-
to'])

```

```

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

    # obtém o ip do usuário
    host = response['ips'][0][1]
    if host == '127.0.0.1':
        host = response['ips'][1][1]

    # título
    texto = font.render('Processador:', True, variaveis['preto'])
    tela.blit(texto, (15, 80))
    # valor
    tela.blit(font.render(processador['nome'], True, variaveis['preto']),
(155, 80))

    # título
    texto = font.render('Frequência:', True, variaveis['preto'])
    tela.blit(texto, (15, 100))
    # valor
    tela.blit(font.render(processador['frequencia'], True, variaveis['preto']),
(155, 100))

    # título
    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))

    # título
    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[ 'pre
to' ]), (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', Tru
e, 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 = cpufreq.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_proce
sso' : 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_dispo
nível, 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():
    """ RESPONSABEL 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():
    """ RESPONSABEL 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():
    """ RESPONSABEL 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: %.2f' % (time.ctime(),
    time.process_time())

    final = time.time() - inicio
    finalClock = time.process_time() - inicioClock

    tempoUsado = 'TEMPO USADO NESSA CHAMADA: %.3f segundos | CLOCK USADO
    NESSA CHAMADA: %.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 ho
    st. ('-c 5') indica, em sistemas linux, que deve mandar 5 pacotes. ('-
    W 3') indica, em sistemas linux, que deve esperar 3 milisegundos por um
    a 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_ele-
mentos_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

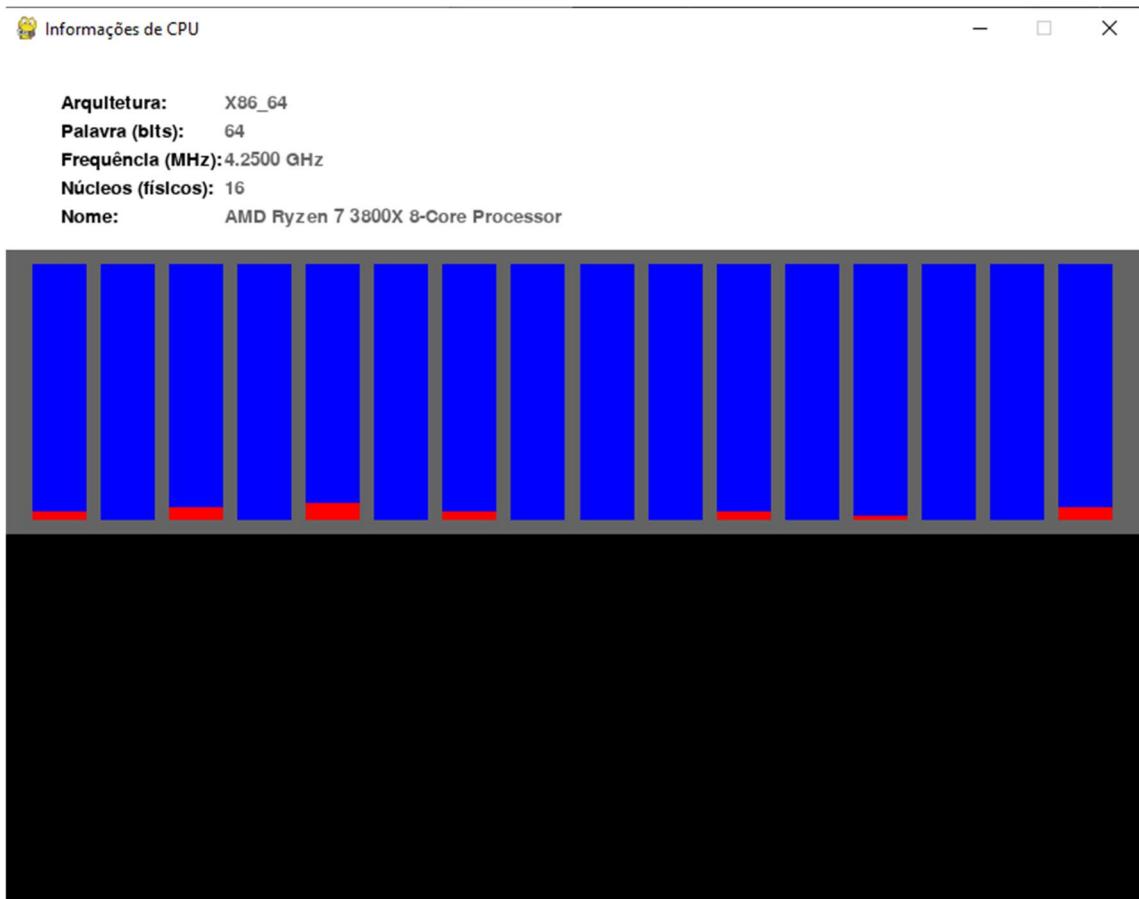


Figure 1 Processador



Figure 2 Memória



Figure 3 Disco



Figure 4 Rede

Evidências v2

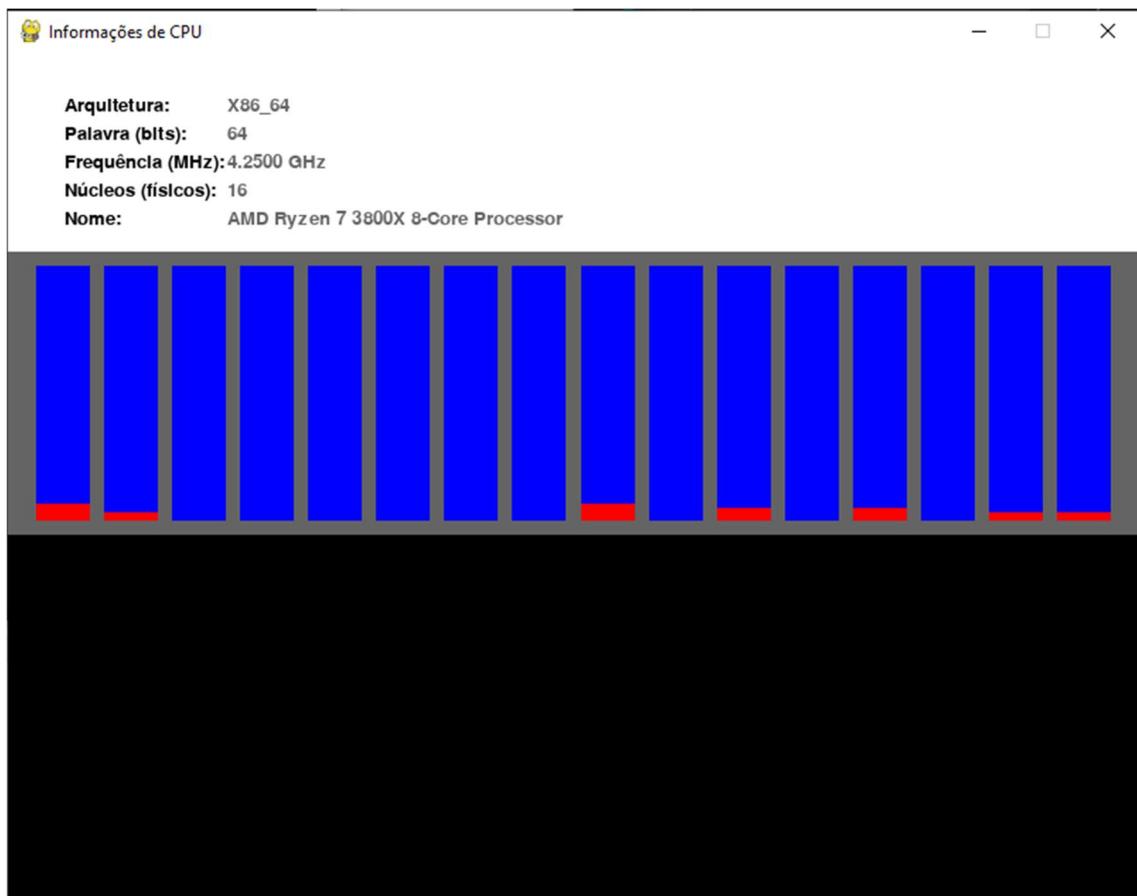


Figure 5 Processador



Figure 6 Memória

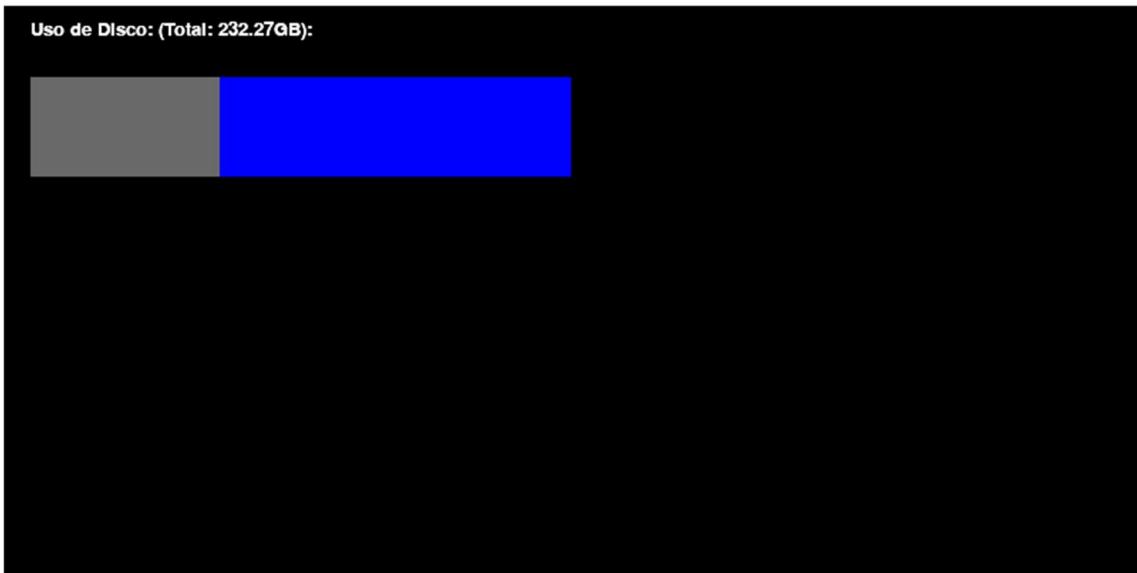


Figure 7 Disco

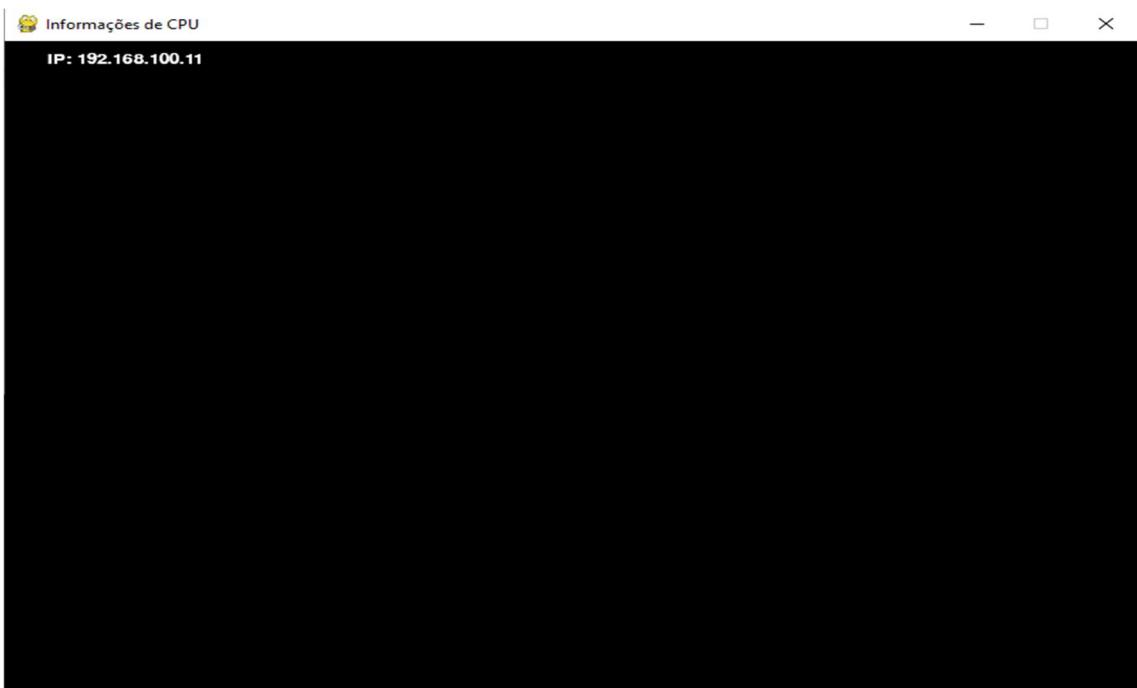


Figure 8 Rede

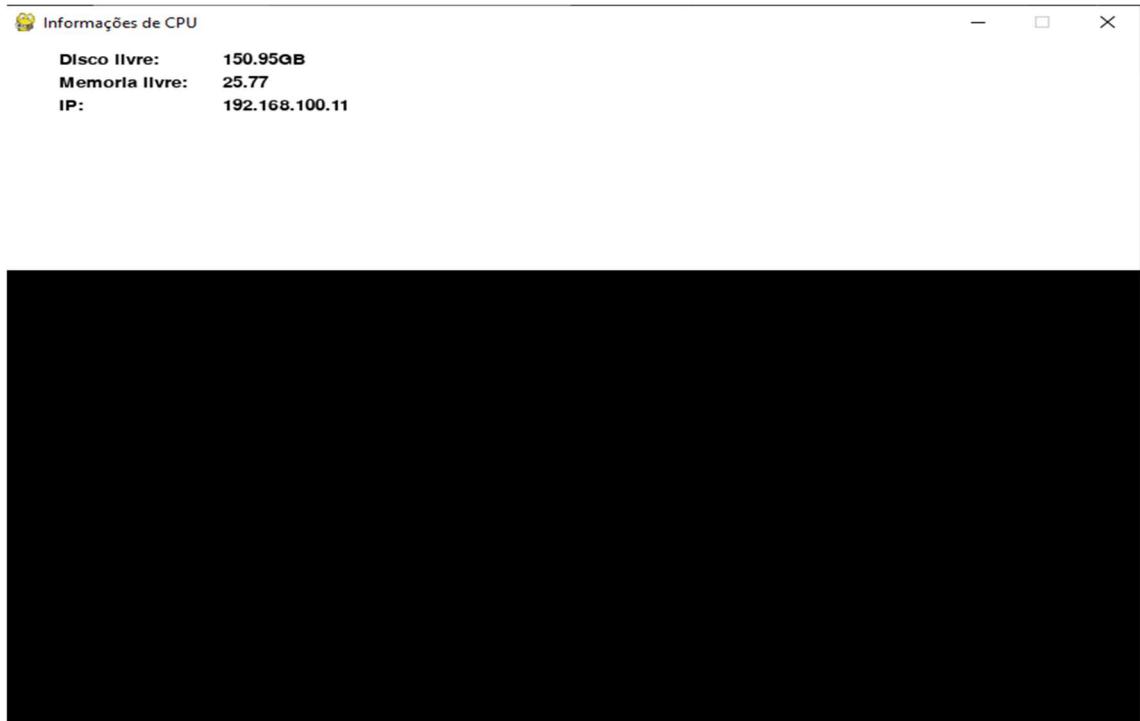


Figure 9 Resumo

Evidências v3

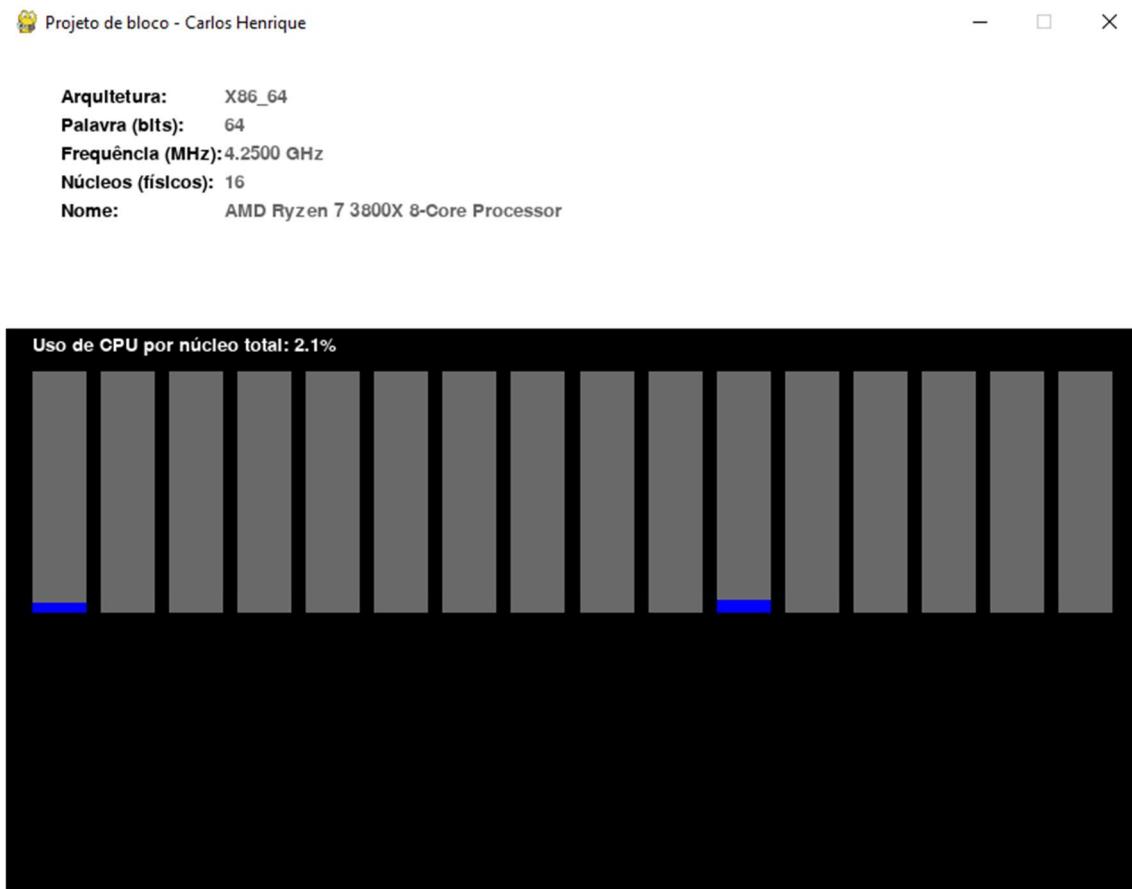


Figure 10 Processador



Figure 11 Memória

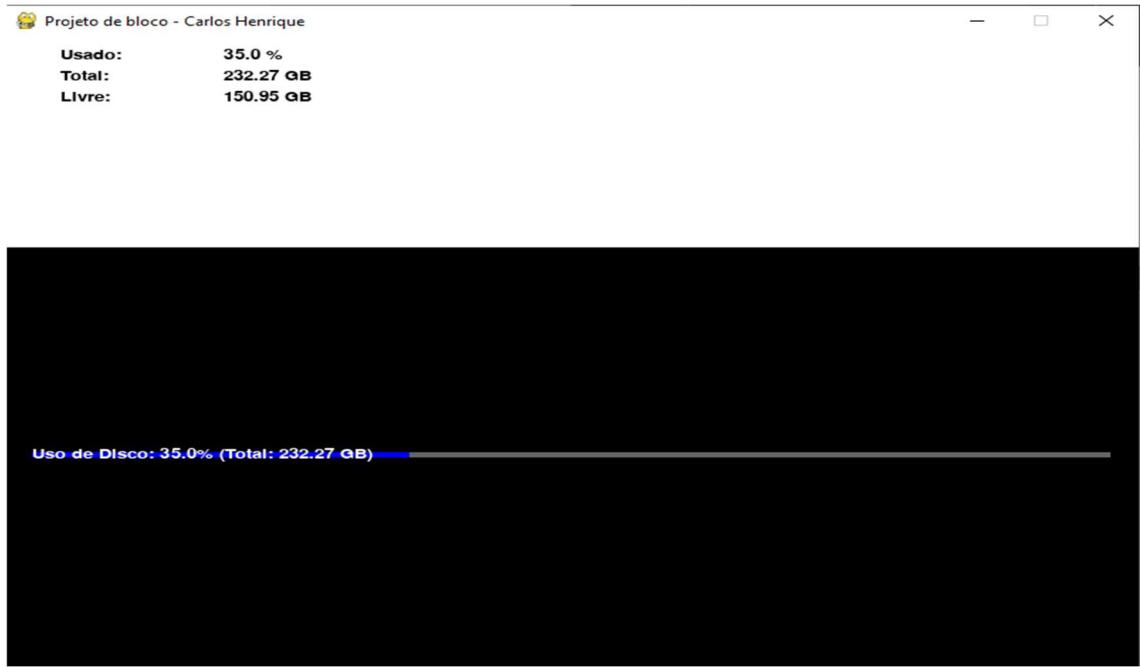


Figure 12 Disco

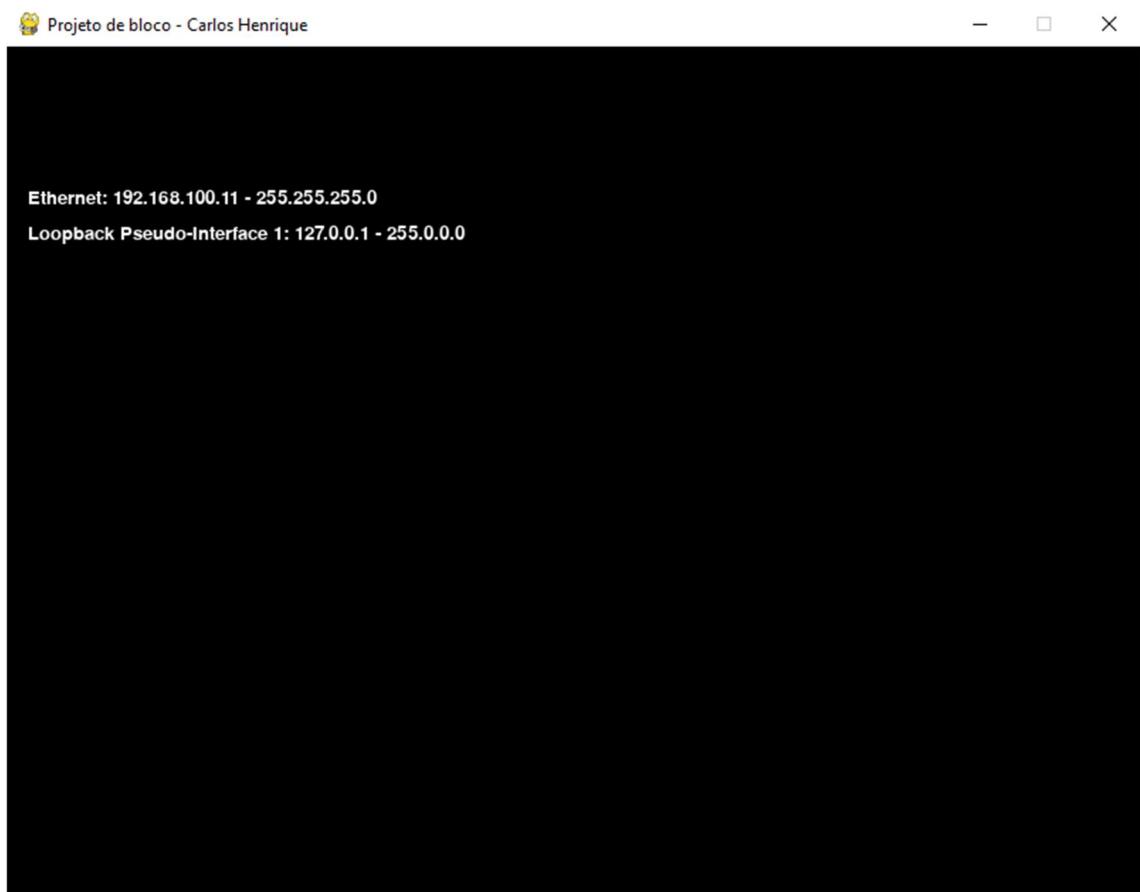


Figure 13 Rede

A screenshot of a Windows File Explorer window titled "Projeto de bloco - Carlos Henrique". The window shows a list of files in a directory. The columns are "Tamanho", "Criação", "Modificação", and "Nome". The files listed are:

Tamanho	Criação	Modificação	Nome
749.60KB	Fri Jan 9 18:13:15 1970	Wed Nov 11 00:51:18 2020	relatorio TP4TP5.pdf
8.81KB	Wed Dec 31 23:30:19 1969	Wed Nov 11 00:51:18 2020	tp2.py
10.00KB	Wed Dec 31 23:50:43 1969	Wed Nov 11 00:51:18 2020	tp3.py
14.02KB	Thu Jan 1 00:59:21 1970	Wed Nov 11 00:51:18 2020	tp4.py
17.77KB	Thu Jan 1 02:03:14 1970	Wed Nov 11 00:51:18 2020	tp5.py

Figure 14 Diretórios

A screenshot of a terminal window titled "Projeto de bloco - Carlos Henrique". The window displays information about a process named "cmd.exe". The output is as follows:

```
Informações sobre processos:  
Nome: cmd.exe  
Executável: C:\Windows\System32\cmd.exe  
Tempo de criação: Wed Nov 11 00:58:10 2020  
Tempo de usuário: 0.0s  
Tempo de sistema: 0.0s  
Percentual de uso de CPU: 1.6%  
Percentual de uso de memória: 0.00%  
Uso de memória: 1.52MB  
Número de threads: 3
```

Figure 15 Processo

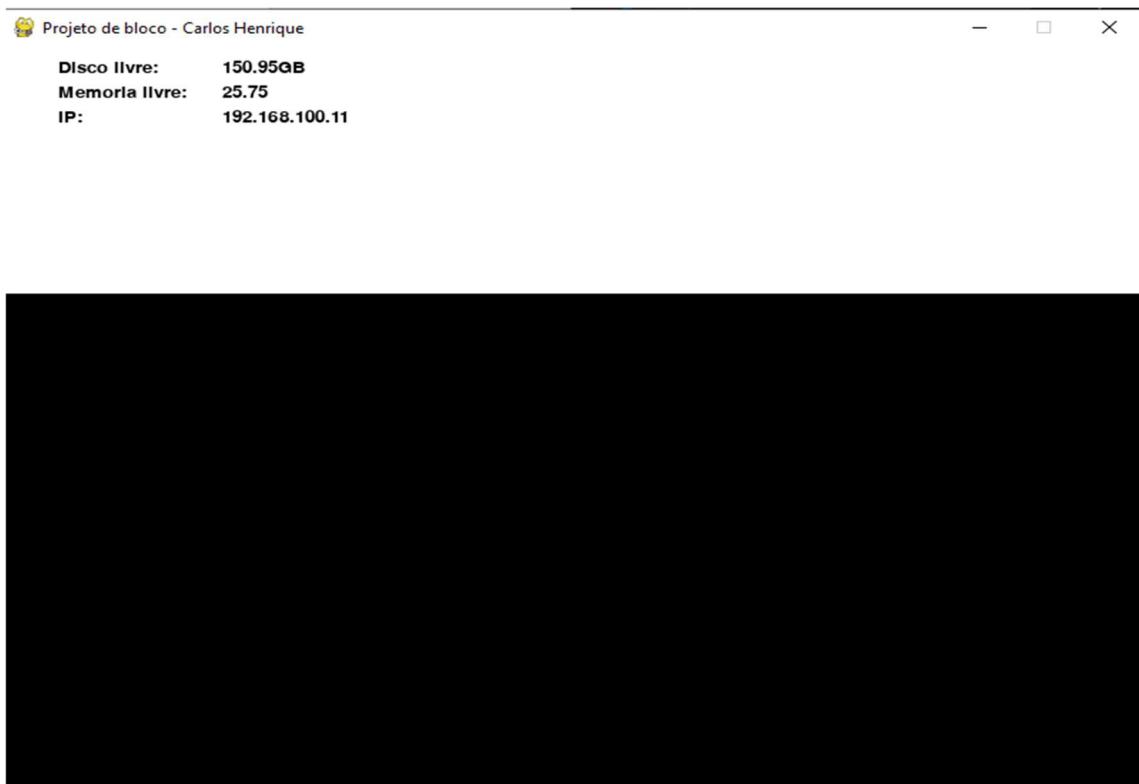


Figure 16 Resumo

Evidências v6



Figure 17 Processador



Figure 18 Memória



Figure 19 Disco

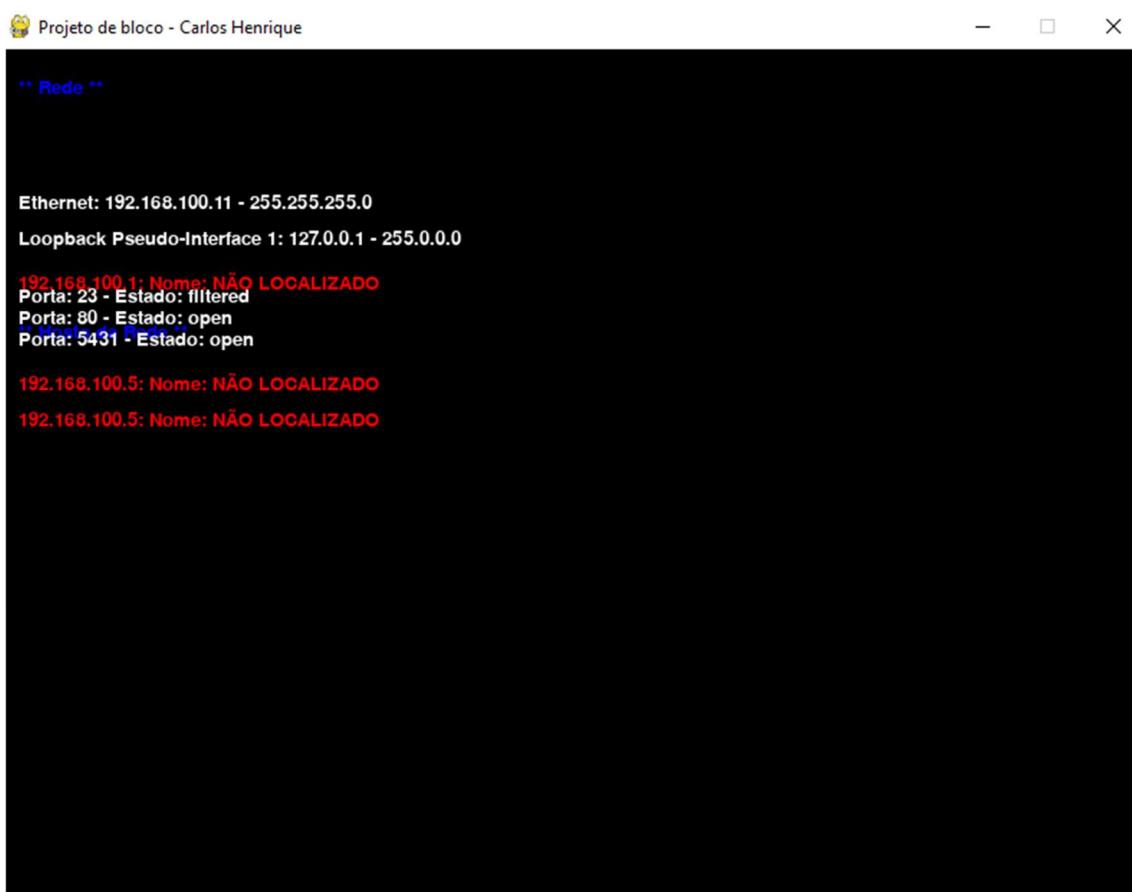


Figure 20 Rede

A screenshot of a Windows File Explorer window titled "Projeto de bloco - Carlos Henrique". The window shows a list of files in a directory. The columns are "Tamanho", "Criação", "Modificação", and "Nome". The files listed are:

Tamanho	Criação	Modificação	Nome
749.60KB	Fri Jan 9 18:13:15 1970	Wed Nov 11 00:51:18 2020	relatorio TP4TP5.pdf
8.81KB	Wed Dec 31 23:30:19 1969	Wed Nov 11 00:51:18 2020	tp2.py
10.00KB	Wed Dec 31 23:50:43 1969	Wed Nov 11 00:51:18 2020	tp3.py
14.02KB	Thu Jan 1 00:59:21 1970	Wed Nov 11 00:51:18 2020	tp4.py
17.77KB	Thu Jan 1 02:03:14 1970	Wed Nov 11 00:51:18 2020	tp5.py

Figure 21 Arquivos

A screenshot of a Windows Task Manager window titled "Projeto de bloco - Carlos Henrique". The window displays information about a running process named "cmd.exe". The details shown are:

Informações sobre processos:

Nome: cmd.exe
Executável: C:\Windows\System32\cmd.exe
Tempo de criação: Wed Nov 11 01:07:43 2020
Tempo de usuário: 0.0s
Tempo de sistema: 0.0s
Percentual de uso de CPU: 1.6%
Percentual de uso de memória: 0.01%
Uso de memória: 1.65MB
Número de threads: 3

Figure 22 Processo



Figure 23 Rede

Evidências v5

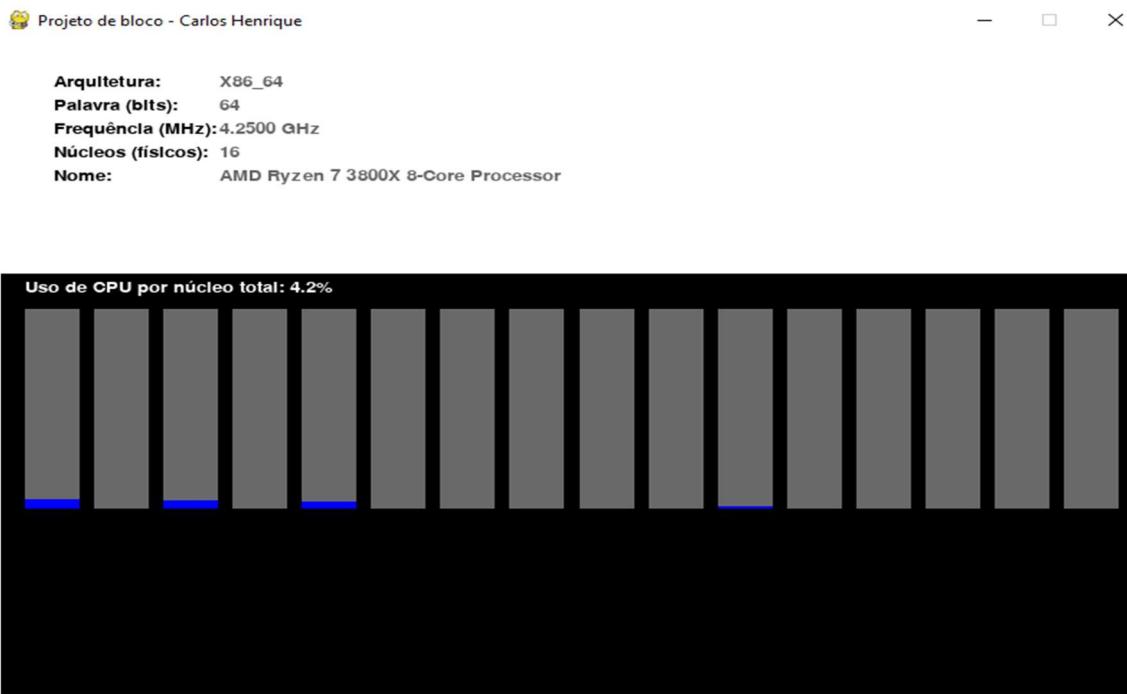


Figure 24 CPU



Figure 25 Memória

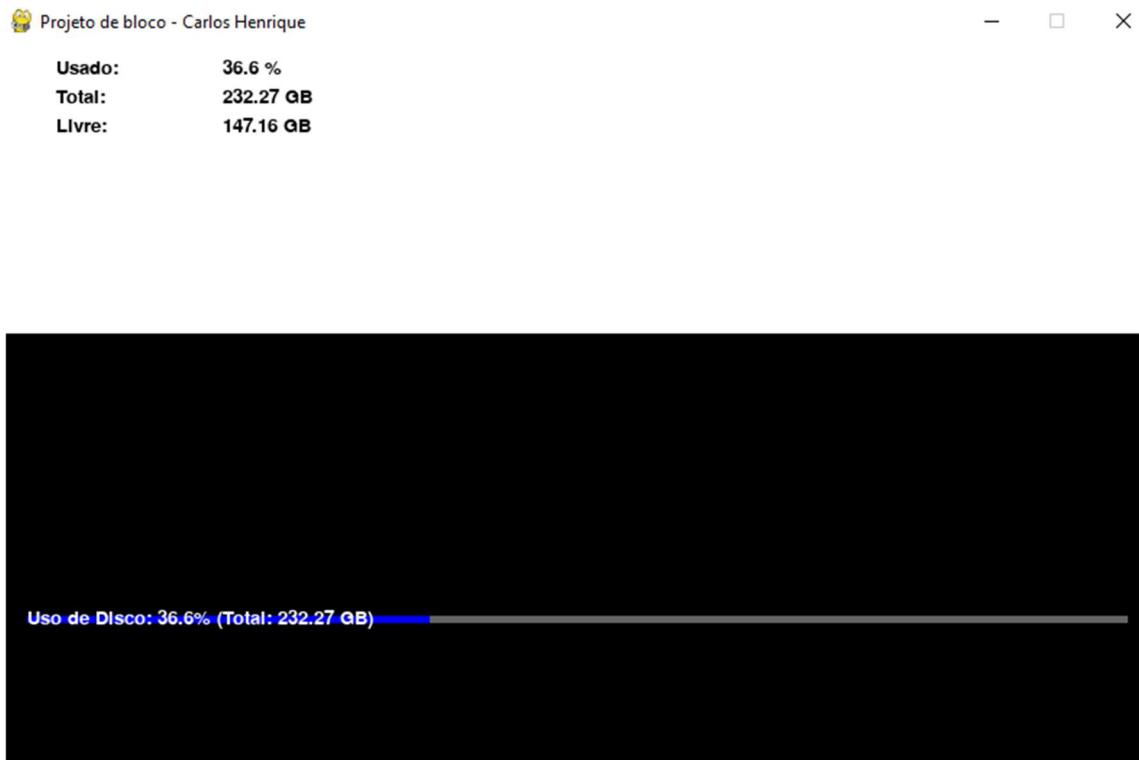


Figure 26 Disco

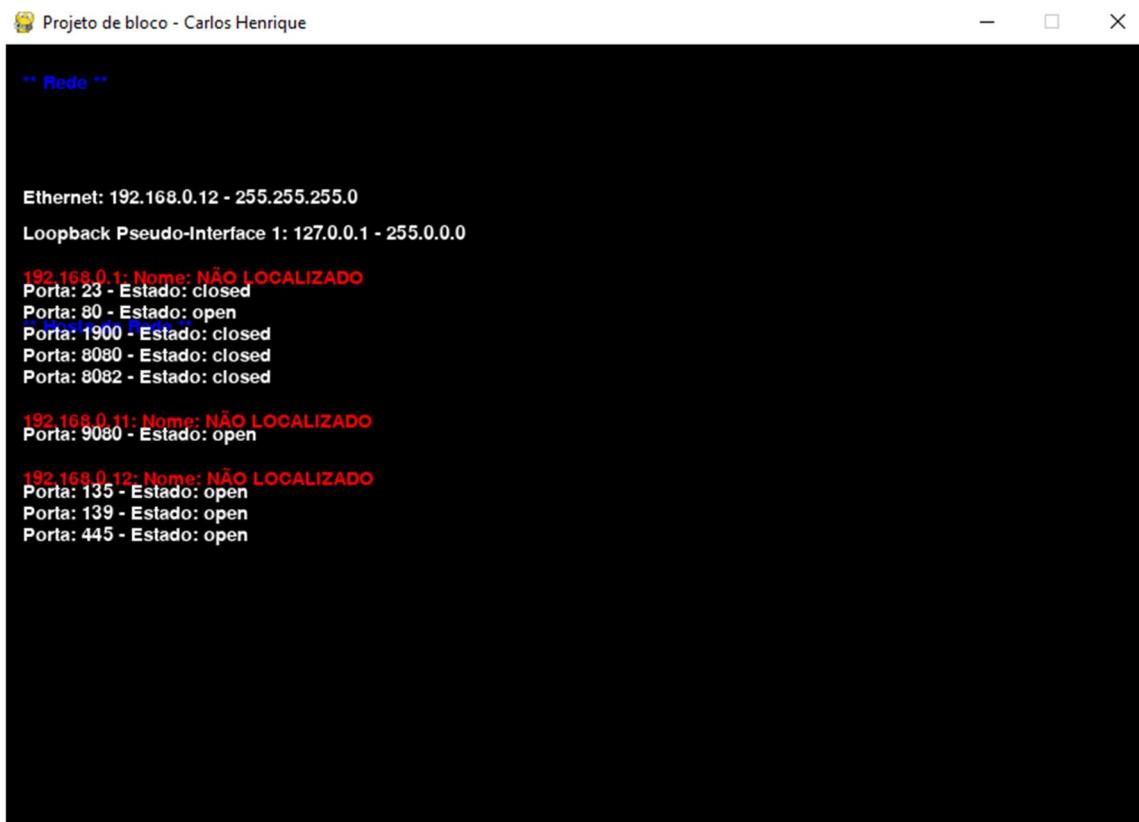


Figure 27 Rede

A screenshot of a Windows File Explorer window titled "Projeto de bloco - Carlos Henrique". The window shows a list of files in a directory. The columns are "Tamanho", "Criação", "Modificação", and "Nome". The files listed are:

Tamanho	Criação	Modificação	Nome
4.00KB	Wed Dec 31 22:08:16 1969	Wed Nov 4 21:29:16 2020	.git
4.00KB	Wed Dec 31 22:08:16 1969	Tue Nov 10 16:20:27 2020	documento
4.00KB	Wed Dec 31 22:08:16 1969	Wed Nov 4 21:29:17 2020	entregavel
726.40KB	Fri Jan 9 11:37:09 1970	Sat Nov 21 00:31:37 2020	log-request.txt
0.01KB	Wed Dec 31 21:00:15 1969	Sat Nov 21 01:09:02 2020	README.md
8.00KB	Wed Dec 31 23:16:32 1969	Wed Nov 4 21:29:17 2020	test
30.87KB	Thu Jan 1 05:46:52 1970	Mon Nov 9 09:55:59 2020	tp-monolito.py
27.58KB	Thu Jan 1 04:50:47 1970	Thu Nov 12 22:36:56 2020	tp8-cliente.py
19.75KB	Thu Jan 1 02:37:02 1970	Thu Nov 12 22:37:05 2020	tp8-server.py

Figure 28 Diretório

A screenshot of a terminal window titled "Projeto de bloco - Carlos Henrique". The window displays system summary information:

Disco Livre:	147.16GB
Memória Livre:	23.34
IP:	192.168.0.12

Figure 29 Resumo

Evidências v6:

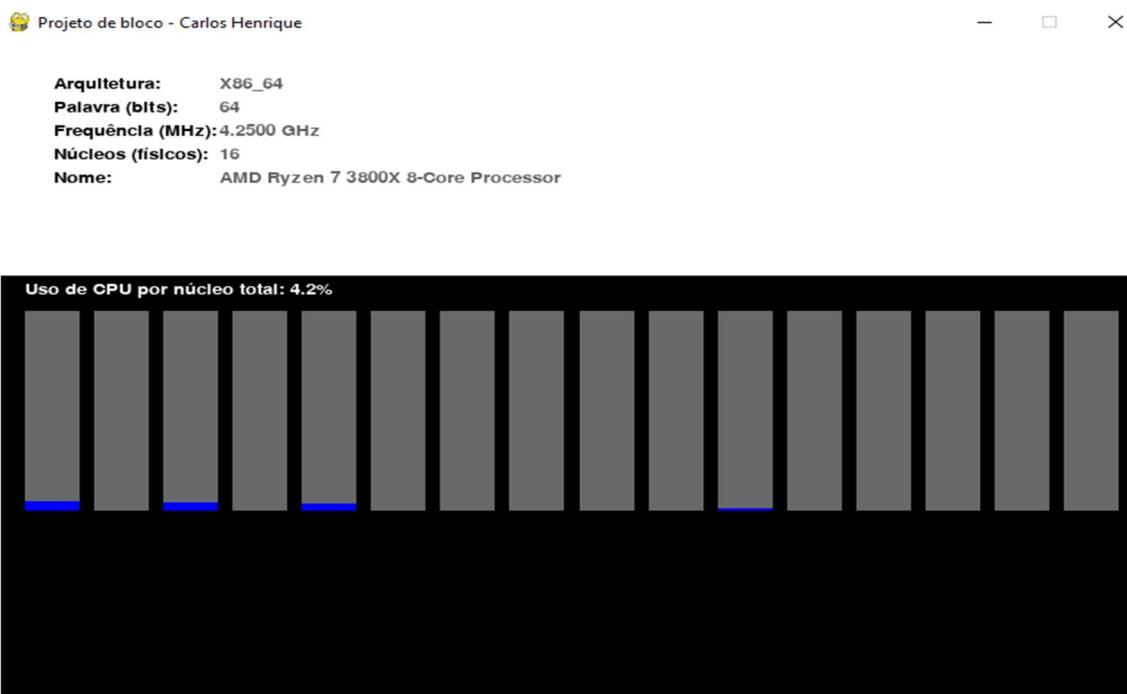


Figure 30 Processador



Figure 31 Memória

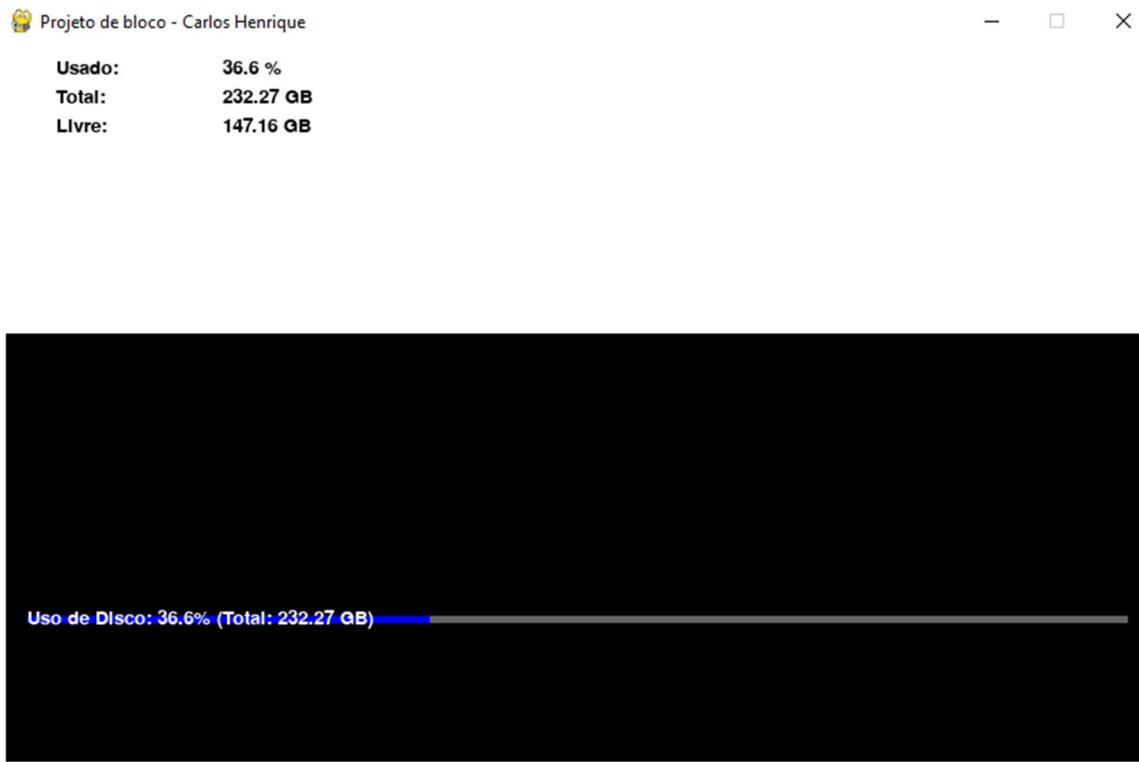


Figure 32 Disco

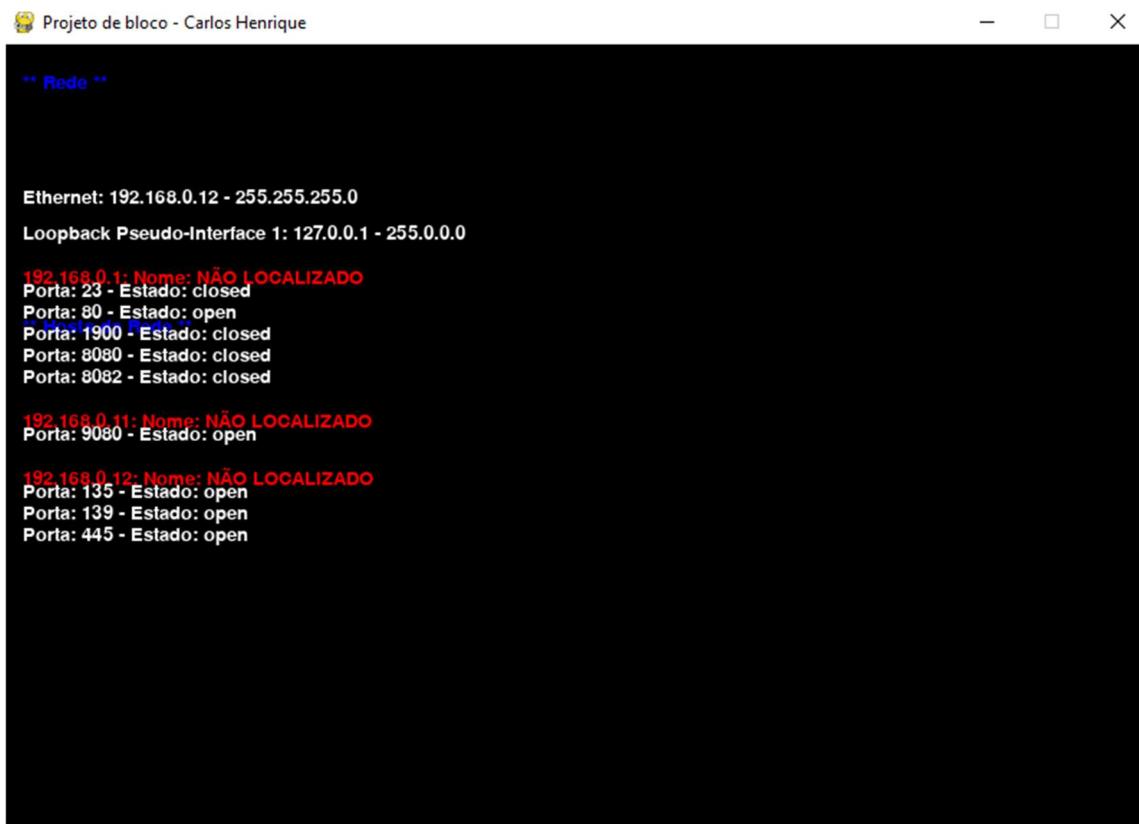


Figure 33 Rede

A screenshot of a Windows File Explorer window titled "Projeto de bloco - Carlos Henrique". The window shows a list of files in a directory. The columns are "Tamanho", "Criação", "Modificação", and "Nome". The files listed are:

Tamanho	Criação	Modificação	Nome
4.00KB	Wed Dec 31 22:08:16 1969	Wed Nov 4 21:29:16 2020	.git
4.00KB	Wed Dec 31 22:08:16 1969	Tue Nov 10 16:20:27 2020	documento
4.00KB	Wed Dec 31 22:08:16 1969	Wed Nov 4 21:29:17 2020	entregavel
726.40KB	Fri Jan 9 11:37:09 1970	Sat Nov 21 00:31:37 2020	log-request.txt
0.01KB	Wed Dec 31 21:00:15 1969	Sat Nov 21 01:09:02 2020	README.md
8.00KB	Wed Dec 31 23:16:32 1969	Wed Nov 4 21:29:17 2020	test
30.87KB	Thu Jan 1 05:46:52 1970	Mon Nov 9 09:55:59 2020	tp-monolito.py
27.58KB	Thu Jan 1 04:50:47 1970	Thu Nov 12 22:36:56 2020	tp8-cliente.py
19.75KB	Thu Jan 1 02:37:02 1970	Thu Nov 12 22:37:05 2020	tp8-server.py

Figure 34 Diretório

A screenshot of a terminal window titled "Projeto de bloco - Carlos Henrique". The window displays system summary information:

Disco Livre:	147.16GB
Memória Livre:	23.34
IP:	192.168.0.12

Figure 35 Resumo

Evidência v7:

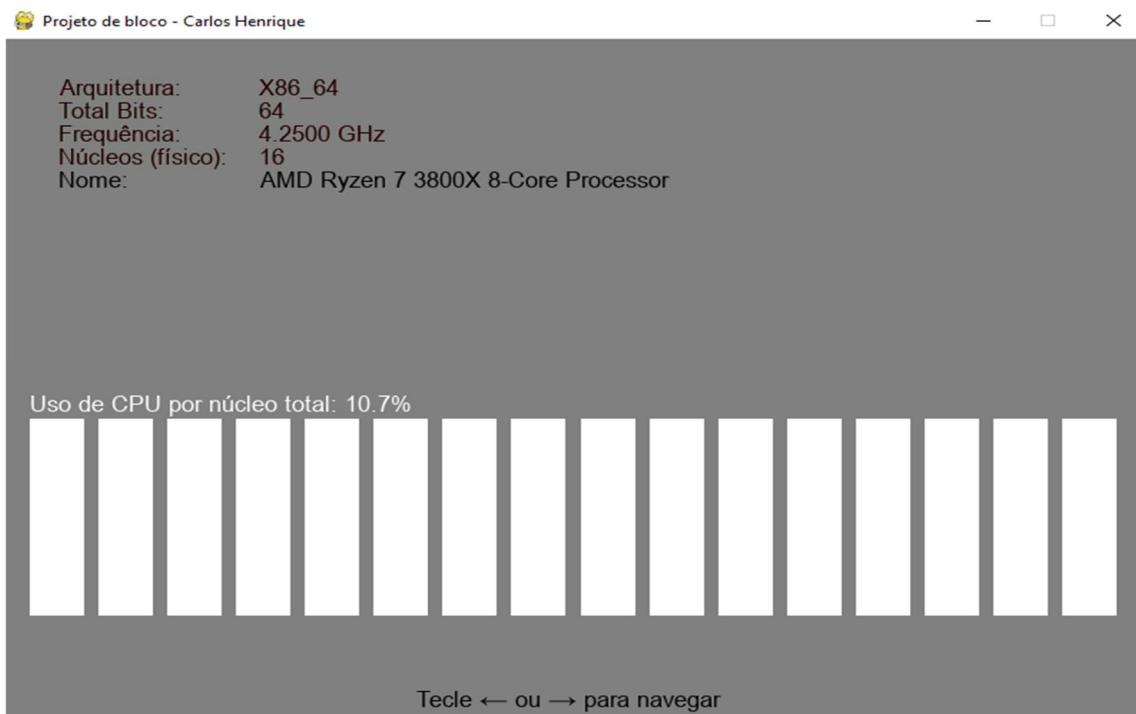


Figure 36 Processador

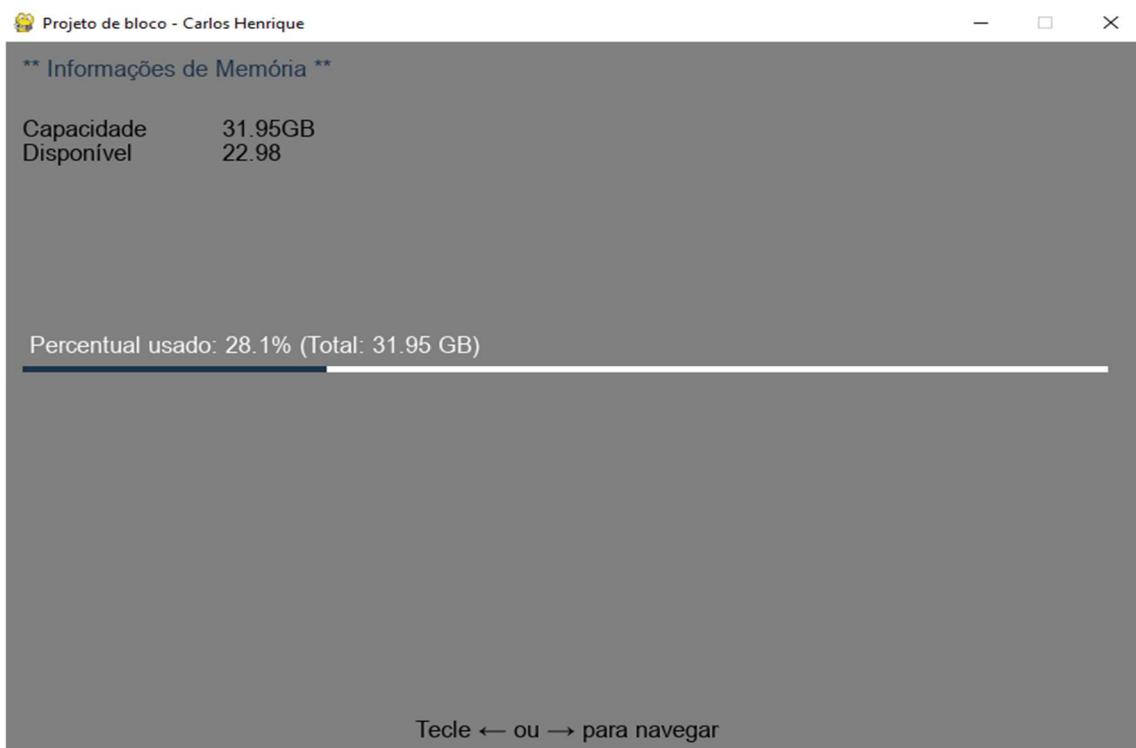


Figure 37 Memória

Projeto de bloco - Carlos Henrique

** Lista dos processos em execução **

PID	% Uso	Mem. Usada	Threads Usada	Tempo	Nome
0	0.00	0B	16.00	0.0 s	System Idle Process
4	0.01	2B	263.00	0.0 s	System
172	0.25	82B	4.00	0.0 s	Registry
600	0.00	1B	2.00	0.0 s	smss.exe
688	0.05	14B	11.00	0.046875 s	python.exe

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40

Tecle + ou - para paginar
Tecle ← ou → para navegar

Figure 38 Processos

Projeto de bloco - Carlos Henrique

** Resumo dos dados coletados **

CPU	
Processador:	AMD Ryzen 7 3800X 8-Core Processor
Frequência:	4.2500 GHz
Bits:	64
<hr/>	
Disco	
Total:	249.4GB
Livre:	158.0GB
Usado:	91.4GB
<hr/>	
Memória	
Total:	31.95GB
Livre:	22.99GB
Usado:	8.96GB
<hr/>	
Rede	
IP:	192.168.0.12

Tecle ← ou → para navegar

Figure 39 Resumo

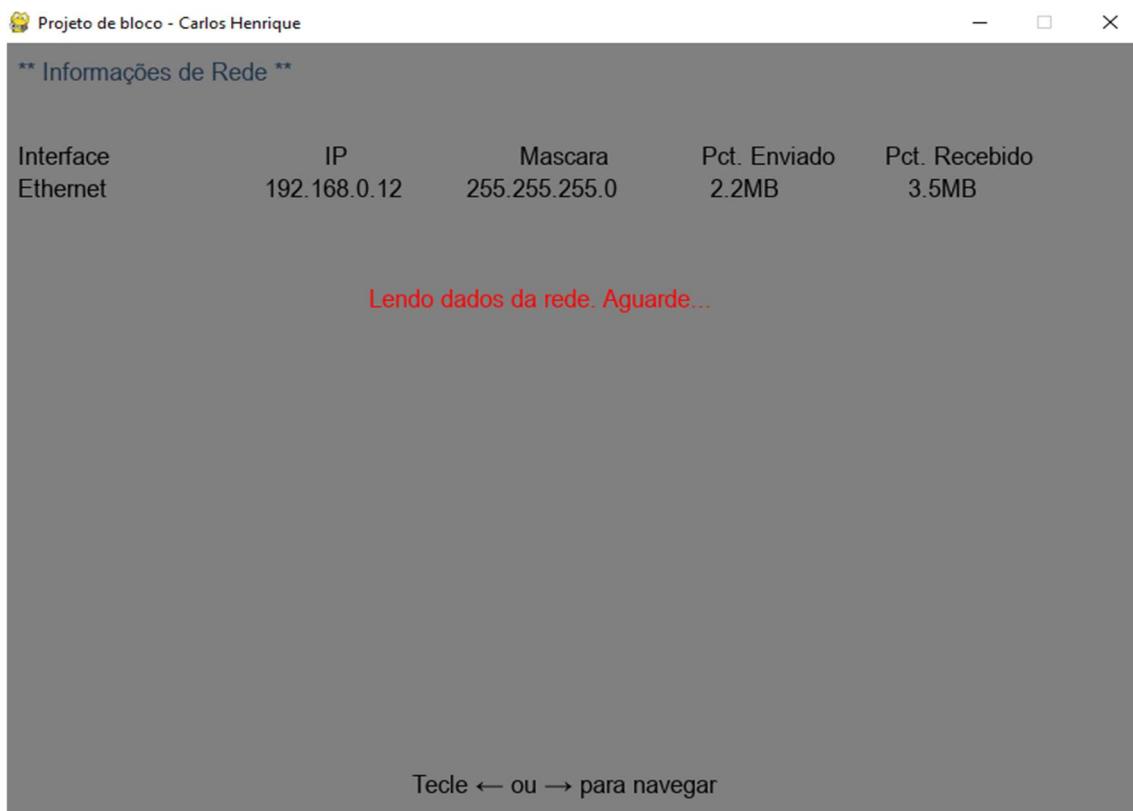


Figure 40 Rede em análise

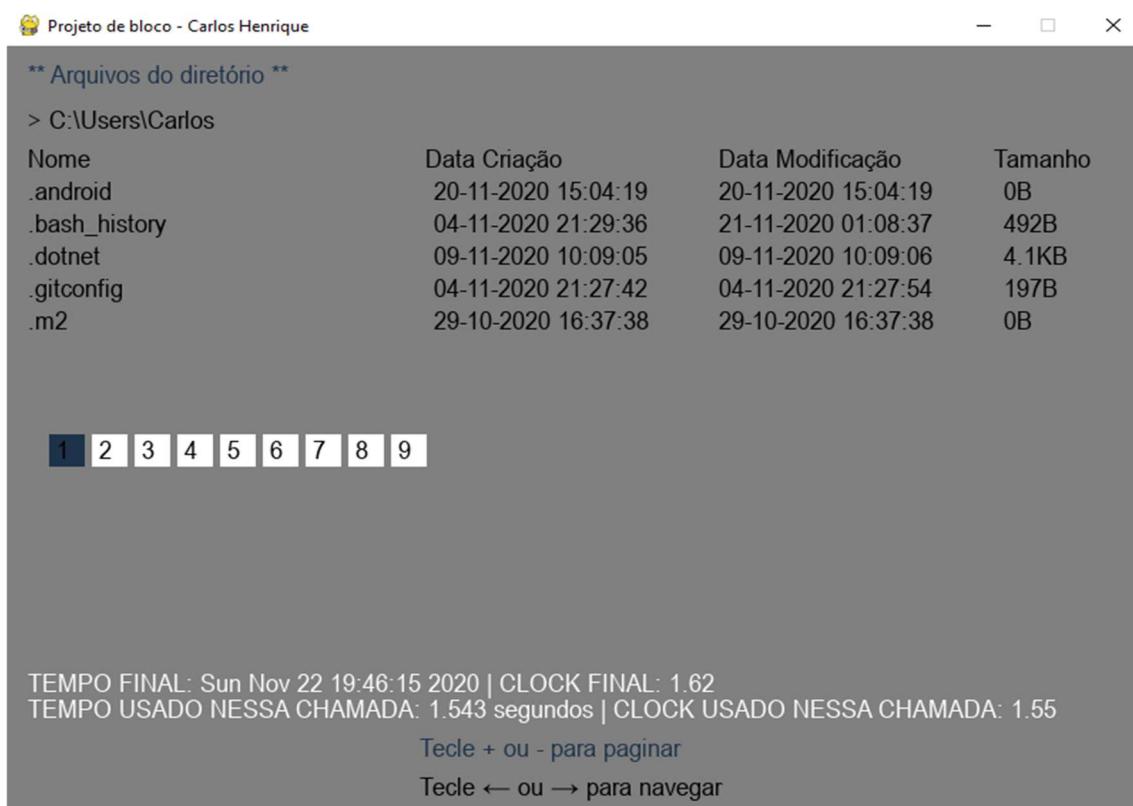


Figure 41 Arquivos

```

Projeto de bloco - Carlos Henrique
** Informações de Rede **

Interface          IP           Mascara      Pct. Enviado   Pct. Recebido
Ethernet          192.168.0.12  255.255.255.0  2.2MB        3.5MB

192.168.0.1: Nome: NÃO IDENTIFICADO
Porta: 23     Estado: closed
Porta: 80     Estado: open
Porta: 1900   Estado: closed
Porta: 8080   Estado: closed
Porta: 8082   Estado: closed

192.168.0.11: Nome: NÃO IDENTIFICADO
Porta: 9080   Estado: open

192.168.0.12: Nome: NÃO IDENTIFICADO
Porta: 135    Estado: open
Porta: 139    Estado: open
Porta: 445    Estado: open
Porta: 9999   Estado: open

Tecle ← ou → para navegar

```

Figure 42 Rede analisada

```

log-request - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
REQUEST: > 2020-11-21 01:00:24.452395 > disco
REQUEST: > 2020-11-21 01:00:24.452395 > disco
REQUEST: > 2020-11-21 01:00:25.748574 > ips
("REQUEST: > ', '2020-11-21 01:00:25.748574', ' ', ' rede')
RESPONSE: > 2020-11-21 01:00:25.748574 > [{"interface": "Ethernet", "enviados": 148872893, "recebidos": 8498451533, "pacotes_enviados": 1261812, "pacotes_recebidos": 7267562}, {"int
dos": 0, "pacotes_enviados": 0, "pacotes_recebidos": 0}], [{"interface": "Ethernet", "enviados": 148879749, "recebidos": 8498460761, "pacotes_enviados": 1261845, "pacotes_recebidos": 0}, {"int
dos": 0, "pacotes_enviados": 0, "pacotes_recebidos": 0}], [{"interface": "Ethernet", "enviados": 148884536, "recebidos": 8498471957, "pacotes_enviados": 1261884, "pac
REQUEST: > 2020-11-21 01:00:25.748574 > trafego
RESPONSE: > 2020-11-21 01:00:25.748574 > [('Ethernet', '192.168.0.12', '255.255.255.0'), ('Loopback Pseudo-Interface 1', '127.0.0.1', '255.0.0.0')]
REQUEST: > 2020-11-21 01:00:27.054763 > ips
REQUEST: > 2020-11-21 01:00:27.054763 > arquivos/1
REQUEST: > 2020-11-21 01:00:28.350942 > ips
("REQUEST: > ', '2020-11-21 01:00:28.350942', ' ', ' rede')
RESPONSE: > 2020-11-21 01:00:28.350942 > [{"interface": "Ethernet", "enviados": 148872893, "recebidos": 8498451533, "pacotes_enviados": 1261812, "pacotes_recebidos": 7267562}, {"int
dos": 0, "pacotes_enviados": 0, "pacotes_recebidos": 0}], [{"interface": "Ethernet", "enviados": 148879749, "recebidos": 8498460761, "pacotes_enviados": 1261845, "pacotes_recebidos": 0}, {"int
dos": 0, "pacotes_enviados": 0, "pacotes_recebidos": 0}], [{"interface": "Ethernet", "enviados": 148884536, "recebidos": 8498471957, "pacotes_enviados": 1261884, "pac
REQUEST: > 2020-11-21 01:00:28.350942 > trafego
RESPONSE: > 2020-11-21 01:00:28.350942 > [('Ethernet', '192.168.0.12', '255.255.255.0'), ('Loopback Pseudo-Interface 1', '127.0.0.1', '255.0.0.0')]
REQUEST: > 2020-11-21 01:00:29.659133 > ips
("REQUEST: > ', '2020-11-21 01:00:29.659133', ' ', ' rede')
RESPONSE: > 2020-11-21 01:00:29.659133 > [{"interface": "Ethernet", "enviados": 148872893, "recebidos": 8498451533, "pacotes_enviados": 1261812, "pacotes_recebidos": 7267562}, {"int
dos": 0, "pacotes_enviados": 0, "pacotes_recebidos": 0}], [{"interface": "Ethernet", "enviados": 148879749, "recebidos": 8498460761, "pacotes_enviados": 1261845, "pacotes_recebidos": 0}, {"int
dos": 0, "pacotes_enviados": 0, "pacotes_recebidos": 0}], [{"interface": "Ethernet", "enviados": 148884536, "recebidos": 8498471957, "pacotes_enviados": 1261884, "pac
REQUEST: > 2020-11-21 01:00:29.659133 > trafego
RESPONSE: > 2020-11-21 01:00:29.659133 > [('Ethernet', '192.168.0.12', '255.255.255.0'), ('Loopback Pseudo-Interface 1', '127.0.0.1', '255.0.0.0')]
REQUEST: > 2020-11-21 01:00:30.970326 > ips
REQUEST: > 2020-11-21 01:00:30.970326 > [{"interface": "Ethernet", "enviados": 148872893, "recebidos": 8498451533, "pacotes_enviados": 1261812, "pacotes_recebidos": 7267562}, {"int
dos": 0, "pacotes_enviados": 0, "pacotes_recebidos": 0}], [{"interface": "Ethernet", "enviados": 148879749, "recebidos": 8498460761, "pacotes_enviados": 1261845, "pacotes_recebidos": 0}, {"int
dos": 0, "pacotes_enviados": 0, "pacotes_recebidos": 0}], [{"interface": "Ethernet", "enviados": 148884536, "recebidos": 8498471957, "pacotes_enviados": 1261884, "pac
RESPONSE: > 2020-11-21 01:00:30.970326 > [{"interface": "Ethernet", "enviados": 148872893, "recebidos": 8498451533, "pacotes_enviados": 1261812, "pacotes_recebidos": 7267562}, {"int
dos": 0, "pacotes_enviados": 0, "pacotes_recebidos": 0}], [{"interface": "Ethernet", "enviados": 148879749, "recebidos": 8498460761, "pacotes_enviados": 1261845, "pacotes_recebidos": 0}, {"int
dos": 0, "pacotes_enviados": 0, "pacotes_recebidos": 0}], [{"interface": "Ethernet", "enviados": 148884536, "recebidos": 8498471957, "pacotes_enviados": 1261884, "pac
<

```

Figure 43 Log