ESTI – ESCOLA SUPERIOR DA TECNOLOGIA DA INFORMAÇÃO
GRADUAÇÃO EM ENGENHARIA DE SOFTWARE
BLOCO DE ARQUITETURA DE COMPUTADORES, SO E REDES

Projeto de Bloco
Arquitetura de Computadores, Sistemas Operacionais e Redes

Fábio Rodrigues Pelagaggi Nunes
Prof.: Adriano Jorge Chame Saad

Rio de Janeiro, 01 de Novembro de 2020.

Sumário

Introdução

Este projeto tem como objetivo a criação de uma ferramenta, utilizando a linguagem de programação Python, que mostre de forma visual a utilização dos recursos de alguns componentes de hardware.

Este software mostra de forma gráfica a porcentagem de utilização da unidade de processamento central, CPU, assim como da memória principal e da unidade principal de armazenamento, assim como algumas outras informações referentes ao hardware do computador.

Para a realização desse projeto foram utilizadas diversas bibliotecas disponíveis para Python, como PYGAME, uma biblioteca que fornece diversos recursos para o desenvolvimento de software multimídia; O PSUTIL, uma biblioteca que fornece recursos para coletar informações do hardware do sistema; Platform, biblioteca nativa Python que também fornece maneiras coletar informações do sistema operacional e hardware.

Graças a essas ferramentas fornecidas pela comunidade de desenvolvedores, o projeto pode ser concluído com sucesso.

Desenvolvimento

O desenvolvimento do projeto foi dividido ao longo de várias etapas, chamadas de versões, onde em cada versão foram implementadas foram novas funcionalidades, realizadas correções de problemas identificados nas versões anteriores e otimizações de implementações das versões anteriores.

Foi escolhido realizar o desenvolvimento do software em inglês, por ser uma língua mundialmente conhecida, e a principal língua utilizada no meio da computação.

A interface do programa foi feita inspirada no Dark Theme do Material Design. Esse estilo de interface foi escolhido por ser um estilo que está ganhando muita popularidade nos últimos anos principalmente por sua simplicidade e eficiência. O Material Design foi desenvolvido pela Google em 2014, para dispositivos Android, e sua popularidade foi tão grande que podemos ver influências desse estilo diversos conteúdos web, não estando mais restrito apenas aos dispositivos Android. O Dark Theme é uma variação, que utiliza cores mais escuras, que vem se tornando cada vez mais comum nos últimos 3 anos, graças a popularização das telas Oleds nos dispositivos móveis.

Versão 1.0

Na primeira versão do programa o objetivo era de criar um programa simples que exibisse de maneira gráfica as seguintes informações do hardware da máquina:

Porcentagem de utilização da CPU.
Porcentagem de utilização da memória principal.
Porcentagem de uso do armazenamento principal.
Informação do IP.

Para implementar essas funcionalidades foram utilizados os módulos Psutil e Platform do Python. Já a interface gráfica foi implementada utilizando o módulo PYGAME.

Logo após importar os módulos mencionados, o desenvolvimento do código foi iniciado realizando a configuração da janela de exibição do programa. Foi configurada uma janela nas dimensões 800 por 600 pixels utilizando a função `pygame.display.set_mode()` do PYGAME. Foi então configurado o loop principal necessário para realizar a atualização das informações exibidas na tela e também necessário para rastrear caso o usuário feche a janela de exibição do programa. Essa checagem das ações do usuário é feita pela função `pygame.event.get()` do PYGAME, com cada ação que o usuário pode tomar tendo que ser prevista e inserida dentro do loop principal, como por exemplo a ação de fechar a janela com a função `pygame.QUIT` do PYGAME. Como pode ser observado abaixo:

```
clock = pygame.time.Clock()
cont = 0
close = False
while not close:
    for event in pygame.event.get():
```

```
        if event.type == pygame.QUIT:
            close = True

    cont += 1
    if cont == 60:
        cpu()
        hd()
        memory()
        pc()
        cont = 0

    pygame.display.update()
    clock.tick(60)
```

Logo após essa primeira configuração, foram definidas duas funções que tem como objetivo preparar as fontes para utilizá-las na exibição do programa. As funções tem como objetivo verificar as fontes instaladas na máquina do usuário e escolher qual deve ser utilizada para exibição. Essa escolha é feita a partir de uma lista, pré definida, com quais fontes devem ser utilizadas em ordem de prioridade.

Primeiro as fontes são inicializadas pela função, `pygame.font.init()`. Com isso, é definida a primeira função `def make_font(fonts, size)` que obtém uma lista com o nome e tamanho de fontes disponíveis na máquina do usuário. Caso tenha uma fonte disponível, a função criará uma instância para a primeira fonte disponível na lista. Se nenhuma fonte da lista estiver disponível, será usada a fonte padrão do sistema. Após essa verificação, a fonte é armazenada em um dicionário `_cached_fonts = {}`.

Com isso, a segunda função `def get_font(font_preferences, size)` é definida. Essa função recebe a seguinte lista com nomes de fontes; `font_preferences = ["Consolas", "Rockwell", "Calibri", "Arial", ]`. E a partir dessa lista, esta função utiliza a função anterior, `make_font(fonts, size)`, e verifica se a primeira fonte da lista está disponível. Caso não esteja, ela passa para próxima fonte da lista. No final, são definidas as fontes e os tamanhos delas que serão usadas no programa.

Com essas configurações iniciais feitas, foi iniciado o processo de configurar as funções principais do código, as responsáveis por para extrair as informações de hardware e exibir na tela.

Essas funções foram definidas acima do loop principal e são apenas chamadas toda vez que exista uma atualização de tela. Como pode ser observado a partir da linha `if cont == 60:` do código acima.

Primeiro foi definido a função para as informações relacionadas a CPU. Utilizando o módulo PSUTIL com a função `psutil.cpu_percent(interval=0.1)` foi adquirido a informação da porcentagem de utilização da CPU, em um intervalo definido de 0.1 segundos. A partir dessa informação é desenhado um retângulo, na cor azul, para representar a utilização de 100% da CPU. Depois é desenhado um outro retângulo, por cima do retângulo azul, mas este vermelho e com dimensões que variam de acordo com a porcentagem de utilização atual da CPU que foi adquirida através da função do PSUTIL. Como pode ser observado a seguir:

```
def cpu():
    cpu = psutil.cpu_percent(interval=0.1)
    width_bar1 = width_window - 2*20
    width_bar2 = ((width_bar1*cpu) / 100)

    pygame.draw.rect(s_cpu, grey_blue, (20, 65, width_bar1, 70))
    pygame.draw.rect(s_cpu, red, (20, 65, width_bar2, 70))
```

O objetivo dessa configuração, é dar uma aparência de um gráfico de barra que varia de tamanho, da direita para a esquerda, de acordo com a utilização da CPU. Ainda utilizando a mesma informação adquirida através da função do PSUTIL, dentro do gráfico de barra, na extremidade esquerda é exibido o número da porcentagem de utilização atual da CPU e na extremidade direita, a porcentagem de CPU não utilizada.

Esta mesma estrutura foi feita para apresentar as informações relacionadas ao uso da memória principal e uso do armazenamento principal, com apenas algumas alterações no código.

Para apresentar as informações relacionadas ao uso da memória principal, foi utilizada a função `psutil.virtual_memory()` do PSUTIL, para fazer a coleta de informações do sistema. Esta função possui diversos atributos, entre eles, foi utilizado 4 atributos; `.total`, retorna à quantidade em bytes de memória total disponível no sistema, `.used`, retorna à quantidade em bytes de memória sendo usada no sistema, `.free`, retorna à quantidade em bytes de memória livre no sistema, `.percent`, retorna a porcentagem de memória utilizada no sistema. Com essas informações coletadas, foi utilizada a mesma estrutura de código da função definida anteriormente `cpu()`, para desenhar as barras representando a porcentagem de uso e exibir as informações correspondentes dentro das barras. No entanto, ao invés de exibir nas extremidades da barra a porcentagem de utilização e a porcentagem não utilizada, foi exibido a quantidade em Giga Bytes de utilização e a quantidade disponível no sistema, como o mostrado abaixo.

```
def memory():
    mem = psutil.virtual_memory()
    memTotalGigas = round((mem.total / (1024*1024*1024)), 2)
    memUsedGigas = round((mem.used / (1024*1024*1024)), 2)
    memFreeGigas = round((mem.free / (1024*1024*1024)), 2)

    barr_text_1 = "Memory Usage"
    bar_text_2 = "(Total: " + str(memTotalGigas) + " GB)"
    bar_text_inside1 = str(memUsedGigas) + " Used"
    bar_text_inside2 = str(memFreeGigas) + " Free"

    text_1 = font_1.render(bar_text_1, 10, grey)
    text_2 = font_1.render(bar_text_2, 10, grey)
    text_inside1 = font_2.render(bar_text_inside1, 10, sand2)
    text_inside2 = font_2.render(bar_text_inside2, 10, dark_sand)

    s_mem.blit(text_1, (20, 10))
    s_mem.blit(text_2, (480, 10))
    s_mem.blit(text_inside1, (35, 110))
    s_mem.blit(text_inside2, (655, 110))
```

As informações referentes ao armazenamento principal foram adquiridas através da função `psutil.disk_usage('/')` do PSUTIL, utilizando os mesmos atributos disponíveis na função para coletar informações da memória principal; `.total`, `.used`, `.free`, `.percent`. Com essas informações a função foi criada de maneira análoga a função `memory()`, sendo definida como `hd()`.

O próximo passo do desenvolvimento, foi a coleta de diversas informações do hardware do usuário. Para isto, foram utilizadas as seguintes funções do módulo Platform: `platform.node()`, para coletar o nome da que está atribuído a máquina do usuário, `platform.system()`, para coletar o nome do sistema operacional, `platform.version()`, para coletar a versão do sistema operacional, `platform.machine()`, para coletar o tipo de arquitetura da máquina, `platform.processor()`, coletar o nome do processador.

Para coletar as duas últimas informações foi utilizado a função `psutil.cpu_count()` para verificar o número de núcleos do processador da máquina, e a função `psutil.net_if_addrs()` para coletar o número de IP da máquina do usuário. Ambas as funções pertencem ao módulo PSUTIL.

Após a coleta de todas as informações elas foram exibidas de forma sequencial, de cima para baixo na janela de exibição.

Primeiro é exibida as informações de uso da CPU, logo abaixo as informações de uso da Memória Principal, abaixo das informações de memória é exibida as informações do uso do Armazenamento Principal, e por último, na parte de baixo da página foi exibida as outras informações a respeito da máquina do usuário, como o nome, sistema operacional, IP entre outras.

Concluiu-se, então, a primeira versão do programa. Pode observar o resultado final nos Anexos I e II, Códigos da Versão 1.0 e Telas da Versão 1.0, respectivamente.

Versão 2.0

A segunda versão do programa tinha como objetivo implementar visualizações alternativas das informações do programa, com cada uma exibindo informações adicionais de hardware. Essas novas visualizações apresentariam as seguintes informações adicionais:

Informações detalhadas sobre o processador.

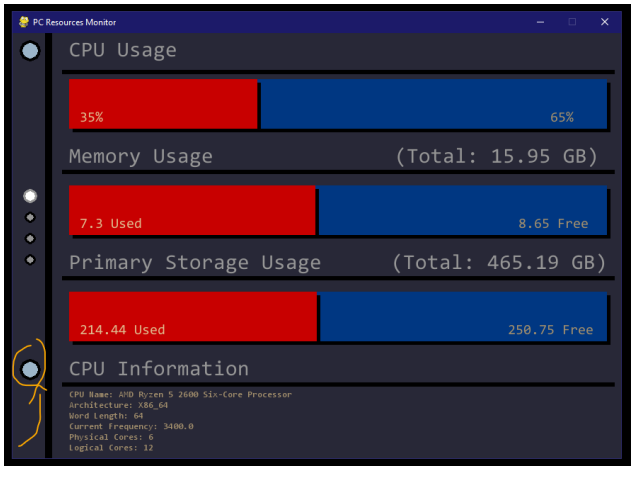
Porcentagem de utilização de cada núcleo da CPU.

As diferentes exibições foram implementadas em forma de carrossel, sendo sua alteração feita por meio das setas do teclado da direita e esquerda, e a barra de espaço para voltar para tela inicial.

Inicialmente foi realizada a coleta das novas informações que seriam exibidas. Utilizando o módulo CPUINFO, foi utilizada a função `cpuinfo.get_cpu_info()`, para coletar diversas informações, organizada em dicionário, a respeito da CPU do sistema. Dessas informações coletadas, foram utilizadas as seguintes informações, adquirida através dos respectivos index; `['brand_raw']`, para coletar o nome comercial do processador, `['arch']`, para coletar a arquitetura utilizada pela CPU e `['bits']`, para coletar o tamanho da palavra da CPU. Foi também utilizada a função `psutil.cpu_freq().current` para fazer o monitoramento da frequência atual de CPU.

Após a coleta das informações, foi configurado uma barra à esquerda da janela, com quatro pontos, para o que o usuário possa acompanhar, de forma visual, qual janela do programa ele está. Também foram adicionados dois botões, um na parte superior, ao lado das informações de uso da CPU, e outro na parte inferior, ao lado das informações gerais da máquina do usuário.

A partir disto, as novas informações adquiridas a respeito da CPU, foram organizadas e exibidas na parte inferior da tela, sendo essa acessada ao trocar de tela, através das teclas de setas para direita e esquerda do teclado, ou através de um botão na parte inferior da barra lateral, como pode ser observado na figura ao lado.



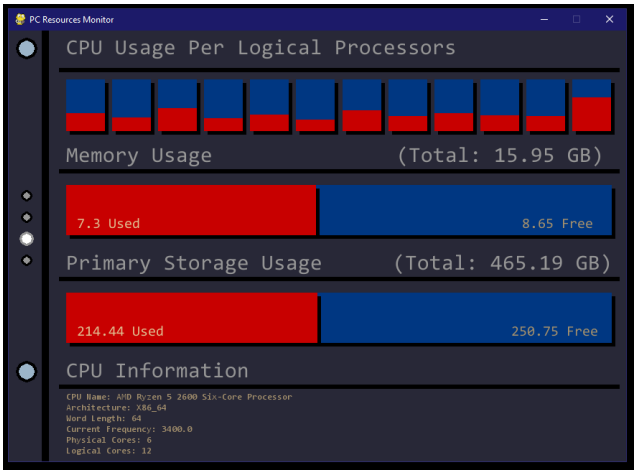
Tela Principal, após pressionar o botão indicado.

O próximo passo foi a implementação de gráficos de colunas para mostrar de forma visual uso de cada núcleo do processador. Para isto, foi criado uma nova função, `cpu_graph_2()` e configurada de maneira similar a função `cpu()`, que foi renomeada para `cpu_graph_1()`. Esta nova função verificar quantos núcleos lógicos existem na máquina do usuário e, a partir disso, desenharia uma coluna azul para cada núcleo, representando o uso 100% do núcleo, e por cima dessa

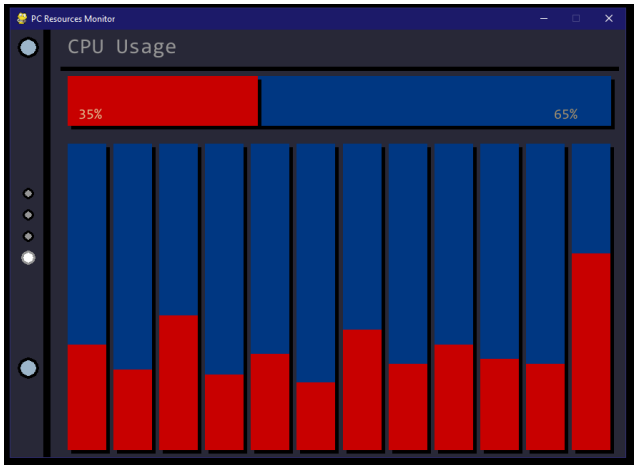
coluna, seria desenhada uma barra vermelha, representando o uso atual do núcleo. Como pode ser observado no código abaixo:

```
def cpu_graph_1():
    cores = len(cores_percent)

    x = y = shift = 10

    height = s_cpu_graph_1.get_height() - 2*y
    width = ( s_cpu_graph_1.get_width() - 2*y - (cores + 1)*shift ) / cores

    d = x + shift

    for i in cores_percent:
        pygame.draw.rect(s_cpu_graph_1, black, (d+5, y+5, width, height))
        pygame.draw.rect(s_cpu_graph_1, red, (d, y, width, height))
        pygame.draw.rect(s_cpu_graph_1, grey_blue2, (d, y, width, ((1-i/100)*height)))

        d = d + width + shift
```

Esse novo gráfico poderia ser acessado através das setas direita ou esquerda, para alterar as páginas de visualização ou através do botão na parte superior da barra lateral esquerda. como mostram as figuras abaixo.

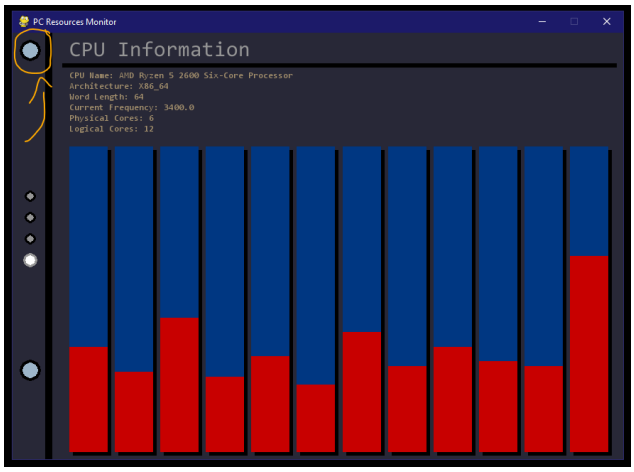


Tela de Principal com gráfico de processamento por Núcleo.



Tela com informações de uso da CPU.

Foi adicionado também a exibição da tela de informações de CPU caso o usuário clique no botão superior da barra da esquerda, estando na tela mostrando o uso de cada núcleo da CPU. Como mostra a figura a seguir:



Tela de CPU, após pressionar o botão indicado.

Concluiu-se, então, a segunda versão do programa. Pode observar o resultado nos Anexos I e II, Códigos da Versão 2.0 e Telas da Versão 2.0, respectivamente.

Versão 3.0

Na terceira versão do programa, foram implementadas diversas correções de bugs, códigos implementados de maneira errada, diversas otimizações na estrutura do código e otimizações na interface visual para conveniência do usuário.

Bugs:

Foram encontrados três bugs na versão anterior.

O primeiro foi que os gráficos com o percentual de uso da CPU e memória não estavam sendo atualizados em tempo real, isto foi causado por um erro na hora de realizar a coleta de informações de dados, que estavam sendo feitas do lado de fora das funções, tornando os dados estáticos uma vez que ao chamar a função as informações que eram utilizadas por ela não eram atualizadas. Isto foi corrigido realizando a coleta das informações dentro das próprias funções, com isso, uma vez que a função era chamada, ela mesma verifica as informações do usuário e desenharia novos gráficos com essas novas informações.

O segundo bug encontrado foi que a função `psutil.cpu_freq().current`, não exibia de maneira correta a frequência da CPU atual no sistema Windows. Este problema foi um pouco mais complexo de resolver, pode-se ver detalhes a respeito deste na sessão O Problema da Frequência, no glossário deste documento. Em resumo, o Windows não tem acesso à informação exata da frequência da CPU, mostrando no gerenciador de tarefas apenas uma estimativa, isso se deve a tecnologia Turbo Boost, que altera a frequência da CPU de acordo com a demanda. Essa tecnologia, comum em todos os processadores atuais, atua em muito baixo nível, sendo completamente fora do controle do Windows. Por causa disso, a função `psutil.cpu_freq().current` não consegue ver a frequência exata da CPU, uma vez que ela usa a referência fornecida pelo OS. Para contornar esse problema foram utilizadas as funções `psutil.cpu_freq().max`, para capturar a frequência máxima da CPU, e `psutil.cpu_percent()`, para capturar a frequência atual de utilização da CPU. Com essas informações foi calculado a frequência atual. Embora não seja a solução ideal, o resultado mostra uma aproximação da frequência atual de utilização de CPU, o que não seria muito diferente da informação fornecida pelo Windows.

O terceiro bug encontrado foi a exibição incorreta das informações de networking da máquina do usuário, que foi corrigido implementando uma nova exibição, para as informações de rede. Utilizando a seguinte função:

```
def get_ip_addresses(family):
    for interface, snics in psutil.net_if_addrs().items():
        for snic in snics:
            if snic.family == family:
                yield (interface, snic.address)

ipv4s = list(get_ip_addresses(socket.AF_INET))
ipv6s = list(get_ip_addresses(socket.AF_INET6))
mac_address = list(get_ip_addresses(psutil.AF_LINK))
```

São coletadas informações sobre IPV4, IPV6 e MAC Address da máquina do usuário em listas contendo tuplas. Com essas informações, é usado um mecanismo para tratar essas informações e exibir para o usuário.

Primeiro é exibido endereço MAC, da máquina do usuário. O endereço MAC, sigla para Media Access Control Address, é um identificador único de cada aparelho que está conectado na rede. É por meio desse identificador que cada aparelho é reconhecido pelo roteador e é atribuído um endereço de IP.

Em seguida é exibido os endereços de IP, da máquina do usuário. O endereço de IP é utilizado, pelo roteador, para realizar a comunicação com a máquina do usuário, e os outros aparelhos que estão na rede. Cada aparelho em uma mesma rede possui um endereço de IP único que é atribuído pelo roteador.

E por fim é exibido os endereços de IPv6 da máquina do usuário. Este endereço funciona da mesma maneira que o endereço de IPv4, exibido anteriormente, no entanto é em hexadecimal o que permite um maior número de computadores na rede.

Otimizações:

Na terceira versão do programa foi feita uma reorganização geral do código, dividindo em 3 arquivos distintos, um para as funções de coleta de dados e exibição das informações da máquina do usuário, chamada data_and_funcs.py, outro para as funções responsáveis pela interface do software, chamada display_interface.py e um último para a função principal, chamada de main.py, no qual chama todos os outros arquivos. Essa organização foi vista como necessário uma vez que o escopo do projeto foi crescendo.

Já na parte da interface visual, foram feitas algumas alterações como de algumas cores para cores menos vibrantes. Foi também removida a barra lateral esquerda, e implementada uma nova barra na parte de baixo da janela, esta nova barra possui um menu interativo com um botão para cada modo de visualização do programa. Foram reduzidos também o número de páginas, exibindo agora menos páginas, mas com uma maior quantidade de informações em cada uma delas.

Implementações:

Nesta terceira versão do programa tinha como objetivo implementar um sistema de que apresente informações sobre o diretório de arquivos como nome, tamanho,

localização, de acesso e data de modificação. Também tinha como objetivo implementar um sistema que apresentem informações sobre os processos do sistema. mostre os processos do usuário, com informações como PID, nome do processo, percentual de uso de CPU e percentual uso da memória principal.

Primeiro foi implementado o sistema para leitura do diretório de arquivos. A leitura de arquivos é feita no diretório de execução do arquivo `main.py,` que foi identificado utilizando a função `os.path.dirname(os.path.realpath(__file__))` e a leitura de arquivos foi feita utilizando a função `os.listdir(local)` ambos do módulo OS. Com esta última, é criada uma lista com o nome de todos os arquivos no diretório, e através desta lista, são as informações de; tamanho de arquivo, data do último acesso, data da última modificação. Utilizando as funções, `.st_size`, `.st_atime` e `.st_mtime` respectivamente. Todas essas informações foram armazenadas em estruturas de dicionários.

Tela de Arquivos exibindo botão.

Tela de Arquivos sem exibir o botão.

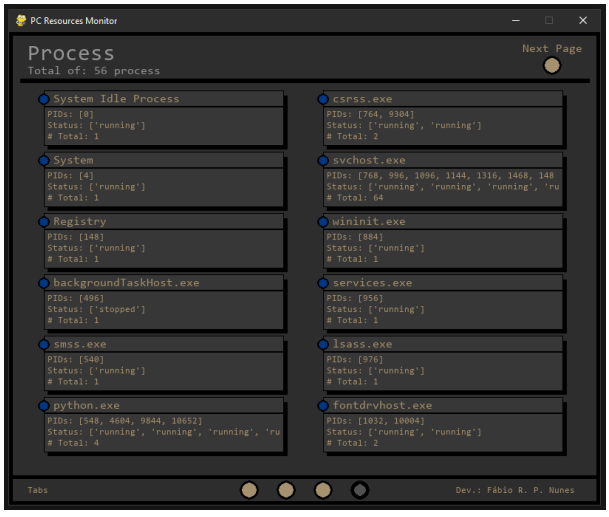
Após a coleta desses dados, foi criada uma outra função com o objetivo de exibir essas informações na tela do programa de forma organizada. Depois, foi implementada uma outra função com o objetivo de verificar se todos os itens foram exibidos na tela e, caso ainda tivesse itens restante é exibido um botão na tela para passar de página, caso contrário o botão não é exibido. O resultado pode ser observado nas imagens abaixo:

Concluída a implementação de leitura de Arquivos, foi implementado a leitura de processos do sistema. A leitura de processos foi feita utilizando a função `psutil.pids()` do PSUTIL, que cria uma lista com os PIDs de todos os processos no sistema operacional. Tendo o PID dos processos, foi criada uma função que lê todos os PIDs da lista, e utilizando as seguintes funções do PSUTIL; `.name()` e `.status()`, foi coletado o nome e o status de cada PIDs e armazenado em uma estrutura de dicionário.

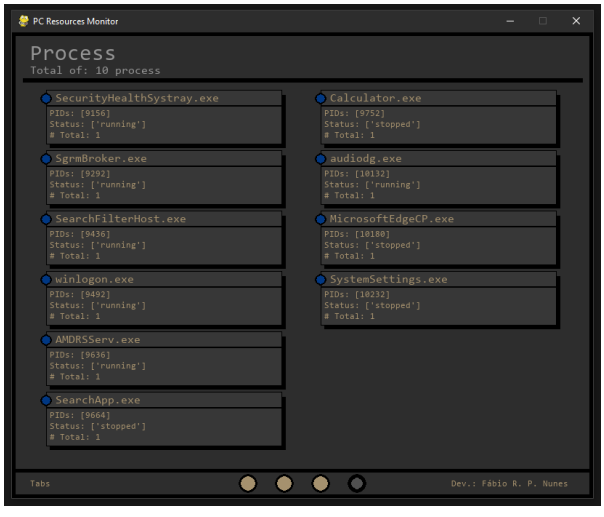
Durante a leitura das informações de nome e status relacionado aos PIDs, foram encontrados alguns problemas. Alguns processos não permitiam a leitura dessas informações, dando a seguinte mensagem de erro: `psutil.AccessDenied: psutil.AccessDenied`. Outros processos, deixavam de existir e no momento de coletar essas informações geraram o seguinte erro: `NoSuchProcess process no longer exists`. A solução encontrada para esse problema foi a cria utilização das estruturas de try e except. Resultando no seguinte código abaixo:

```
for i in processes:
        try:
            proc_name = str(psutil.Process(i).name())
            if proc_name not in process:
                process[proc_name] = {}
                process[proc_name]['PIDs'] = []
                process[proc_name]['Status'] = []
                process[proc_name]['# Total'] = 0
            process[proc_name]['PIDs'].append(i)
            process[proc_name]['Status'].append(psutil.Process(i).status())
            process[proc_name]['# Total'] += 1
        except (psutil.AccessDenied, psutil.ZombieProcess):
            pass
        except psutil.NoSuchProcess:
            continue
```

Para a exibição de informações foi adotada a mesma estrutura utilizada anteriormente, para a exibição dos arquivos. Como pode ser observado abaixo:



Tela de Processos sem exibir o botão.



Tela de Processos exibindo botão.

A coleta das informações de processos é uma operação que demanda um alto desempenho, por isso é fica evidente um grande atraso na resposta do programa ao clicar no botão para próxima página, parecendo às vezes que o programa parou de responder.

Concluiu-se, então, a terceira versão do programa. Pode observar o resultado nos Anexos I e II, Códigos da Versão 3.0 e Telas da Versão 3.0, respectivamente.

Versão 4.0

A quarta versão do programa tinha como objetivo realizar testes de performance nas diversas funções do software, e com isso verificar quais as funções que estão utilizando uma maior clock da CPU e realizar otimizações.

Implementações:

Foi implementado um sistema de chamadas às funções `time.perf_counter()` do módulo TIME, para medir o tempo de execução de todos os processos em cada página do software. O sistema consiste em uma chamada a função `time.perf_counter()` no início do loop principal da função `main()` e uma outra chamada após a execução de todas as funções em cada uma das páginas de exibição do programa. Foi também implementado um contador para mudar de página automaticamente, sem a necessidade de um input do usuário.

Foram também realizados testes utilizando a função `time.sleep()` que, suspendendo os processos por alguns segundos, esses segundos continuam sendo contados ao utilizar a função `time.perf_counter()`, o que não ocorria com a antiga função `time.clock()` e também não ocorre com a função `time.process_time()`. A função `time.perf_counter()` foi usada ao invés da função `time.process_time()` por ter uma maior precisão nos resultados dos segundos, o que é o ideal para testes de performance. Após os testes foi obtido o seguinte resultado abaixo:

```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Process Time on Home Page: 0.20591140000000002
Process Time on Home Page: 0.20115399999999994
Process Time on Home Page: 0.20139549999999995
Process Time on Home Page: 0.20240570000000013
Process Time on Home Page: 0.2019358
Process Time on CPU Page: 2.5636587000000004
Process Time on CPU Page: 2.5533552999999998
Process Time on CPU Page: 2.5684553999999995
Process Time on CPU Page: 2.5662705
Process Time on CPU Page: 2.5567690999999986
Process Time on FILES Page: 0.0024537999999996174
Process Time on FILES Page: 0.0017863999999985225
Process Time on FILES Page: 0.0023149000000000086
Process Time on FILES Page: 0.0020060000000015066
Process Time on FILES Page: 0.0020550999999997543
Process Time on PROCESS Page: 0.6314056000000008
Process Time on PROCESS Page: 0.6253834000000005
Process Time on PROCESS Page: 0.626665599999999
Process Time on PROCESS Page: 0.6294025999999988
Process Time on PROCESS Page: 0.6283194999999999
```

Observando os resultados dos testes, é possível notar uma maior demanda do processador ao executar as funções da página CPU, atingindo uma demanda de processamento podendo ultrapassar 100 vezes quando comparado com a página de menor demanda de processamento, FILES. Essa alta demanda da página da CPU era

esperado, uma vez que as funções estão constantemente verificando percentual de uso de cada núcleo do processador e exibindo o resultado em um gráfico de colunas.

Outra página com uma alta demanda de processamento foi a página de processos, PROCESS. Esse resultado também era esperado, uma vez que as funções estão constantemente verificando quais processos estão sendo executados, e coletando informações a respeito deles.

Após esse resultado foi realizado testes utilizando o módulo SCHED, para atribuir diferentes prioridades as funções da página CPU, que demandava um maior processamento. A página CPU utiliza 3 funções para sua exibição, sendo assim foi utilizado a função `scheduler.enter()`, cada função recebeu uma prioridade diferente, colocando uma maior prioridade na função de maior demanda e menor prioridade nas funções de menor demanda.

```
                scheduler.enter(0, 2, data_and_funcs.cpu_graph_0)
                scheduler.enter(0, 1, data_and_funcs.cpu_graph_1)
                scheduler.enter(0, 3, data_and_funcs.cpu_inf)
                scheduler.run()
                t2 = time.perf_counter()
                print("Process Time on CPU Page: " + str(t2 - t1))
                counter_change_page += 1
                if counter_change_page == 5:
                    display_interface.menu = 'FILES'
                    counter_change_page = 0
```

Foi obtido o seguinte resultado:

```
Process Time on CPU Page: 2.5634433999999997
Process Time on CPU Page: 2.555506000000001
Process Time on CPU Page: 2.560194299999999
Process Time on CPU Page: 2.5556915999999994
Process Time on CPU Page: 2.5629582000000006
```

Pode-se observar que o resultado foi muito semelhante ao anterior. Este resultado pode indicar que o problema da alta demanda das funções não é relacionado com as prioridades de execução das funções, mas pode ser relacionado com a demanda de execução da própria função.

Foi então implementado o módulo THREADING, do Python. O objetivo é atribuir uma threading para a execução de cada função da página CPU de observar se com isto, haveria um ganho em performance. A implementação e seus resultados podem ser vistos abaixo:

```
                thrad_1 = threading.Thread(target=data_and_funcs.cpu_graph_0)
                thrad_2 = threading.Thread(target=data_and_funcs.cpu_graph_1)
                thrad_3 = threading.Thread(target=data_and_funcs.cpu_inf)
                thrad_1.start()
                thrad_2.start()
                thrad_3.start()
                thrad_1.join()
                thrad_2.join()
                thrad_3.join()
                t2 = time.perf_counter()
```

```
                print("Process Time on CPU Page: " + str(t2 - t1))
```

Resultando:

```
Process Time on CPU Page: 1.3619709999999996
Process Time on CPU Page: 1.3558462999999996
Process Time on CPU Page: 1.3561572999999996
Process Time on CPU Page: 1.3580098999999999
Process Time on CPU Page: 1.3562179000000008
```

Desta vez, o resultado obtido é muito diferente dos anteriores, mostrando um ganho de desempenho de mais de 50%. O que indica que ao realizar a execução das funções de forma paralela, existe um ganho de performance significativo, e o tempo elevado de execução da página provavelmente é devido a execução da função de maior demanda, `cpu_graph_1()`.

Futuras Implementações:

No momento, ao ir para a página de arquivos ou processos, caso o limite de exibição tenha sido ultrapassado, apenas um botão para a próxima página é exibido. Uma vez que se muda a página, não é possível retornar a página anterior. Uma das implementações que poderá ser realizada nas próximas versões será de um botão para retornar à página anterior.

Outra implementação possível é a de exibição de maiores informações a respeito dos processos no sistema. Exibindo o uso de memória e CPU de cada processo.

Concluiu-se, então, a quarta versão do programa. Pode observar o resultado nos Anexos I e II, Códigos da Versão 4.0 e Telas da Versão 4.0, respectivamente.

Versão 5.0

A quinta versão do programa tinha como objetivo implementar um sistema para realizar uma verificação da rede no usuário. Verificando quais IPs estão sendo utilizados na rede e extrair algumas informações dos IPs.

Bugs:

Apesar de ser mencionado na Versão 1.0 o programa não exibia a porcentagem de utilização da memória e do armazenamento principal. Tinha apenas as barras para representar a porcentagem de utilização e a informação da quantidade, em gigas, de utilização de cada um deles. Foi, então, implementada a exibição, em falta, da porcentagem de utilização de cada um desses recursos. Sendo estes exibidos na barra, logo acima da informação da quantidade de utilização. A implementação foi feita utilizando a função `.percent` do PSUTIL.

Outro problema encontrado foi que por vezes a página Files e a página Process, por vezes não respondia ao clicar para mudar para a próxima página. Este problema era mais evidente na página Process, pela alta demanda de processamento exigida para exibição das informações. Para tentar solucionar este problema foi implementado um novo sistema, completamente diferente, para realizar a paginação das informações exibidas. Nesse novo sistema foram implementados dois botões dinâmicos para passar para a página seguinte ou retornar à página anterior. Esses botões só são exibidos para o usuário caso exista uma página seguinte ou alguma página pra retornar.

Otimizações:

Com um novo sistema de paginação implementado, foi realizado testes de performance para observar se existe algum ganho de performance ao aplicar o uso de Multithreading em ambas as páginas.

Foi desta vez, foi obtido o seguinte resultado sem o uso de Multithreading na página Process:

```
Process Time on PROCESS Page: 2.536089
Process Time on PROCESS Page: 2.4690331
Process Time on PROCESS Page: 2.4546535
Process Time on PROCESS Page: 2.4220165
Process Time on PROCESS Page: 2.4124626000000013
Process Time on PROCESS Page: 2.3973701
Process Time on PROCESS Page: 2.3954432000000008
Process Time on PROCESS Page: 2.3895690999999992
```

E o seguinte resultado foi obtido após o uso de Multithreading na página Process:

```
Process Time on PROCESS Page: 2.3861567999999984
Process Time on PROCESS Page: 2.3929287000000024
Process Time on PROCESS Page: 2.3653367000000003
Process Time on PROCESS Page: 2.3569907999999984
Process Time on PROCESS Page: 2.358047400000004
Process Time on PROCESS Page: 2.3591095999999965
Process Time on PROCESS Page: 2.3567417000000006
Process Time on PROCESS Page: 2.3622145000000003
```

Com estes resultados, pode-se se concluir que não houve ganho significativo com o uso de Multithreading na página Process, no entanto o programa pareceu responder melhor a paginação com sua aplicação.

Por motivo de performance, foi implementado um novo sistema para coleta de informações dos processos ativos na máquina do usuário. Com esse novo sistema os processos do usuário seriam coletados apenas uma vez, na inicialização do software, ao invés de ser atualizado uma vez a cada 6 segundos como estava anteriormente. Com esse novo sistema houve um grande ganho em performance, como pode ser observado nos tempos de processamento da página mostrado abaixo, quando comparado com o tempo de processamento anterior.

```
Process Time on PROCESS Page: 0.0035755000000001758
Process Time on PROCESS Page: 0.00355210000000028
Process Time on PROCESS Page: 0.003791000000001432
Process Time on PROCESS Page: 0.0035791999999990054
Process Time on PROCESS Page: 0.0037591999999992964
Process Time on PROCESS Page: 0.0036613000000009777
Process Time on PROCESS Page: 0.0035210999999986115
Process Time on PROCESS Page: 0.003509499999999832
```

Implementações:

Foi implementado um sistema para verificar todos os IPs da rede e retornar quais IPs estão sendo utilizados. Esse sistema foi implementado utilizando a função `subprocess.call()` do módulo SUBPROCESS. Esta função foi utilizada para fazer o sistema operacional realizar um Ping em todos os IPs da sub rede do usuário, do número 0 até o número 255. E com isto foi verificado quais pings retornaram, os IPs que retornaram os pings foram guardados em uma lista.

Após obter a lista com todos os IPs ativos da rede foi utilizado a função `nmap.PortScanner()` do módulo NMAP do Python, para coletar as informações dos IPs da rede do usuário.

Tendo todas as informações, elas foram exibidas em uma quinta páginas, nomeada de ‘Networking Informations’. Nesta página todos os IPs são exibidos, seguido das informações de MAC, Vendor e o tipo de resposta que foi obtido de cada IP. Como pode ser observado abaixo:

Tela de Networking exibindo informações dos IPs da rede.

Futuras Implementações:

No momento em que se inicia o programa começa a realizar a verificação de todos os IPs da rede e coletar suas informações. É somente após esse processo ser concluído que o programa se torna disponível para a utilização do usuário. Foram realizadas diversas tentativas para conseguir realizar a coleta dessas informações em background enquanto o restante do programa está disponível para o usuário. Foram realizados testes utilizando diversas variáveis de controle, em conjunto com Multithreading e o módulo Queue, um módulo nativo do Python para criar filas de processos. Mas todos esses testes não tiveram sucesso.

Outras maneiras de realizar essa implementação estão sendo estudadas e serão testadas para a implementação nas próximas versões.

Concluiu-se, então, a quinta versão do programa. Pode observar o resultado nos Anexos I e II, Códigos da Versão 5.0 e Telas da Versão 5.0, respectivamente.

Versão 6.0

A sexta versão do programa tinha como objetivo implementar um sistema que realize uma verificação do tráfego de rede na máquina do usuário. Esse sistema tinha como principal objetivo verificando todo o tráfego realizado por cada interface de rede do usuário, e verificar todo o tráfego de rede realizado por cada processo na máquina do usuário.

Implementações:

Foi implementado um sistema para realizar o monitoramento do tráfego de todas as interfaces de rede disponíveis na máquina do usuário. Esse monitoramento é realizado utilizando a função `psutil.net_io_counters()` do PSUTIL. Essa função retorna um dicionário contendo todas as interfaces de rede como index, seguido de uma tupla contendo todas as informações de rede desta interface.

Após a coleta desses dados, essas informações foram tratadas, colocando em estruturas indexadas de dicionários para facilitar a manipulação dos dados, e exibidas em uma nova página do software nomeada de ‘Traffic Informations’. Nesta página todas as interfaces são exibidas, seguida das informações de tráfego destas interfaces, bytes enviados, bytes recebidos, quantidade de pacotes enviados e quantidade de pacotes recebidos.

Concluída esta etapa, foi implementado um sistema de monitoramento das conexões realizadas por processos. Para isto foi utilizada a função `.connections()` do PSUTIL, esta função retorna uma tupla com informações de rede do processo. Foi implementada uma função para checar cada processo do sistema operacional e armazenar e coletar as seguintes informações; status da conexão, endereço local, porta local, endereço remoto e porta remota.

Após a coleta dessas informações, elas foram exibidas na página ‘Traffic Informations’ seguido das informações de tráfego da interface.

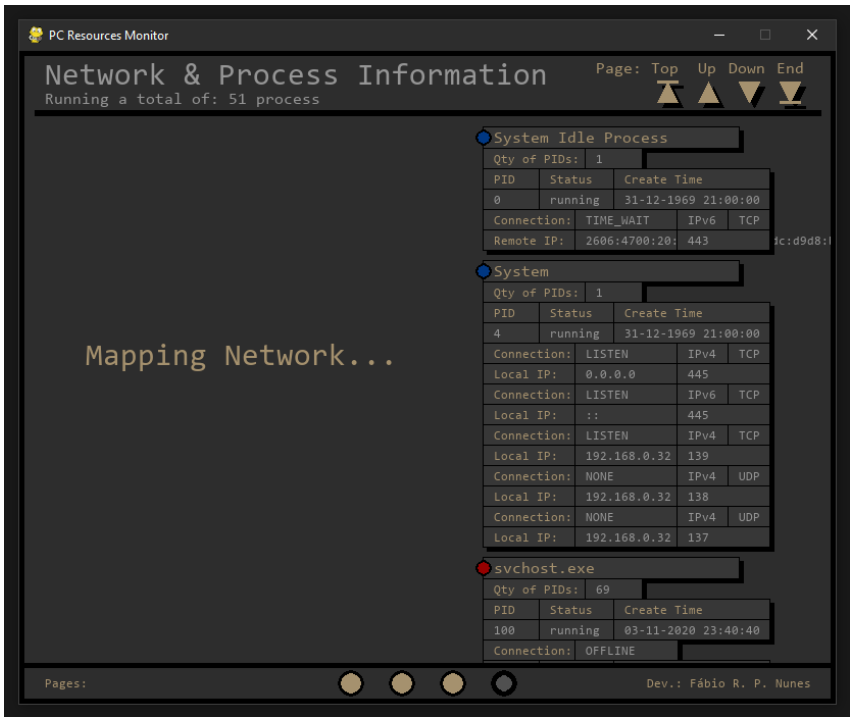
Concluiu-se, então, a sexta versão do programa. Pode observar o resultado nos Anexos I e II, Códigos da Versão 6.0 e Telas da Versão 6.0, respectivamente.

Versão 7.0

A sétima versão do programa tinha como objetivo transformar o software em uma ferramenta cliente / servidor. Para isto, foi necessário realizar a reestruturação de algumas funções, alterando a maneira que alguns dados eram coletados e exibidos para o usuário. Foi realizada, também, uma nova organização do código nos arquivos. E criado um arquivo, chamado de server.py, responsável em armazenar todas as funções necessárias para a execução do servidor.

Bugs:

Como mencionado na versão 6.0 do software, durante essa versão era necessário aguardar o final do mapeamento da rede para que o software fique disponível para a utilização do usuário. Durante a implementação do mapeamento, foram testadas sem sucesso diversas maneiras para realizar esse mapeamento em paralelo, deixando o programa disponível para o usuário.



Página Network & Process, Mapeamento com Multithreading

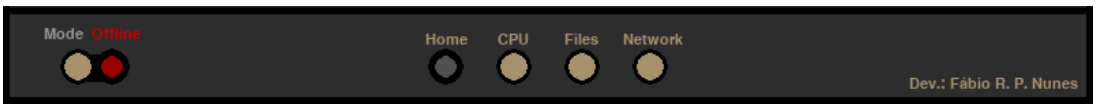
Após novas tentativas, foi implementada com sucesso o mapeamento de rede em paralelo, deixando o restante do software disponível enquanto o mapeamento é realizado. Para realizar essa implementação foram necessários conhecimentos avançados do módulo Multithreading do Python, em especial a função `threading.Thread()`. Para realizar o mapeamento da rede foram utilizadas 3 funções principais. As duas primeiras são descritas na versão 5.0 do programa.

Já a terceira foi criada com o objetivo de retornar chamar as duas outras funções e adicionar os resultados dessas a uma lista, e o Multithreading foi aplicado a esta terceira função. Isto foi necessário devido a maneira de como o Multithreading se comporta no sistema. Quando uma função é executada utilizando o Multithreading, não é possível retornar nenhum valor dela, e não é possível alterar nenhuma variável global. Com essas limitações foi necessário criar duas listas vazias como variáveis globais e, dentro da função com Multithreading, apenas adicionar os resultados a essas listas já previamente criadas. Com esta implementação é possível deixar o restante do programa em execução

enquanto a thread preenche as listas com as informações do mapeamento de rede. Como pode observar na imagem ao lado.

Otimizações:

Para proporcionar uma melhor experiência para o usuário, foi expandida a barra de menu na parte de baixo da tela, com isto foi inserido um rótulo em cima de cada botão das páginas do software, tornando assim a sua utilização mais eficiente, uma vez que o usuário já sabe qual o botão para ativação de cada página. Como pode ser observado abaixo:



Barra de Menu

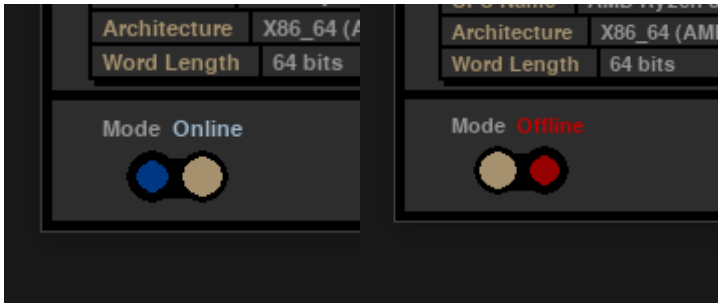
Com o objetivo de otimizar a visualização de informações nas telas, foi realizada a união das páginas, Network, Traffic e Process em apenas uma nova página. Esta página, chamada de Network & Processes Informations, reúne as informações de todas as páginas anteriores. Foram movidas para esta nova página as informações de rede que eram exibidas na página inicial do programa, e, para evitar informações redundantes e fora de contexto, as informações exibidas na página inicial do programa foi alterada para exibir somente as informações a respeito do hardware do usuário.

Para a exibição da nova página Network & Processes Informations, foi adotado um sistema de rolamento de cima para baixo, que pode ser feito utilizando os botões no canto superior direito da tela, ou utilizando as teclas; PgUp, para rolar a página para cima; PgDown, para rolar a página para baixo; Home, para ir direto para o topo da página; End, para ir direto para o final da página. A rolagem também pode ser feita utilizando o botão giratório do meio do mouse.

Com o objetivo de padronizar os elementos que são exibidos na tela, facilitar a organização e proporcionar uma mesma experiência a todos os usuários, foram retiradas as diferentes fontes de exibição. Deixando assim apenas a fonte padrão do PYGAME para todos os usuários.

Implementações:

Para realizar a implementação do software em cliente / servidor, foi adicionado uma nova opção na barra de menu do software que altera o modo de operação do mesmo para online ou offline. No modo online, o programa tenta realizar uma conexão com o servidor, caso a conexão seja bem sucedida, o modo é alterado de offline para online, caso não seja bem sucedida, o modo continua a mostrar como offline. Por padrão o software é inicializado no modo offline, sendo necessário clicar no botão do menu, ou pressionar a tecla TAB, para mudar para o modo online. Como mostra as figuras abaixo:



Modo Online *Modo Offline*

Como mencionado anteriormente, nesta versão do programa, foi feita uma reorganização em todo o código. Nesta nova estrutura, muitas funções que realizavam a coleta de informações da máquina do usuário e exibição, receberam um novo parâmetro que diz se o software está online ou offline, caso estiver online, a função utiliza as informações passadas como parâmetro para a exibição das informações para o usuário, e caso estiver offline, a função coleta os dados para exibição na tela. Essa implementação foi utilizada em todas as funções responsáveis por realizar a exibição de informações no software, pode observar um exemplo desta implementação, abaixo, na função de exibição de informações de memória:

```
def mem_graph(connection, mem_data):
    # Data
    if connection == 'ONLINE':
        mem = mem_data
    else:
        mem = psutil.virtual_memory()

    mem_total = round((mem.total / (1024*1024*1024)), 2)
    mem_used = round((mem.used / (1024*1024*1024)), 2)
    mem_free = round((mem.free / (1024*1024*1024)), 2)
    mem_percent = int(mem.percent)
```

A implementação do servidor foi realizada utilizando a função `socket.socket()` do módulo SOCKET do Python, sendo passado como parâmetros tipo de IP, IPv4 ou IPv6, e de conexão, UDP ou TCP, tendo sido passadas como parâmetros. Em seguida, foi vinculado o endereço IP e uma porta, ao socket criado, utilizando a função `socket.bind()`, e colocado o servidor em espera, aguardando uma conexão, utilizando a função, `socket.listen()`. Caso alguma tentativa de conexão ao socket é realizada, esta conexão é aceita através da função `socket.accept()`.

Com a conexão realizada, a troca de informações entre cliente e servidor foi feita utilizando o módulo PICKLE, do Python, por ele fornecer ferramentas de conversão de estruturas complexas, como listas e dicionários, por exemplo, para bytes. Sua utilização foi essencial, para o desenvolvimento do desta etapa, uma vez que todos os dados coletados são trabalhados e organizados nessas estruturas, e a comunicação cliente servidor só realizada em binário.

Para converter os dados para binário, foi utilizada a função `pickle.dumps()`, e para converter novamente para dados, do lado do cliente, foi utilizada a função `pickle.loads()`, ambas do módulo PICKLE.

Após os dados serem convertidos para binários, a troca de informações foi realizada utilizando as funções `socket.send()`, para o envio, e `socket.recv()` para o recebimento, ambas do módulo SOCKET.

Com essa estrutura estabelecida, foi implementado um sistema em que o cliente envia informações referentes à qual página o usuário se encontra, e com isso o servidor envia os dados necessários para aquela página específica. Desta maneira quando uma conexão for estabelecida com o servidor, todo o programa exibe dados referentes ao servidor conectado.

Uma nova estrutura de arquivos foi adotada, nesta o código foi separado em 5 arquivos. 4 arquivos para o cliente; main.py, que contém o loop principal do software, é responsável por reunir as informações dos outros arquivos e possuí diversas variáveis de controle necessárias para a execução do programa. interface.py, contém todas as informações e funções que define a aparência da interface do programa. functions.py, contém todas as funções do programa responsável por realizar a coleta de dados e exibição, mas que não são relacionadas aparência da interface. database.py, armazena todas as informações que são coletadas da máquina do usuário, utilizando os diversos módulos do Python e as funções definidas no arquivo functions.py. E 1 arquivo para o servidor; server.py, contendo todas as funções e coleta de dados necessárias para a execução de um servidor.

Concluiu-se, então, a sétima versão do programa. Pode observar o resultado nos Anexos I e II, Códigos da Versão 7.0 e Telas da Versão 7.0, respectivamente.

Módulos

Neste capítulo apresentaremos todos os módulos utilizado durante o projeto com uma breve explicação das funções utilizadas.

Informações completas sobre cada um desses módulos poderá ser encontrada na sua documentação oficial, os links estão disponíveis na seção Bibliografia.

CPUINFO

Cpuinfo é um módulo que tem como objetivo obter informações da CPU com Python puro. Py-cpuinfo deve funcionar sem quaisquer programas ou bibliotecas extras, além de funcionar em sistemas Linux, OS X, Windows, BSD, Solaris, Cygwin, Haiku e BeagleBone.

As seguintes funções desse módulo foram utilizadas no software:

`cpuinfo.get_cpu_info()` - Recolhe uma lista contendo diversas informações da CPU, que podem ser acessadas a partir das chaves.
Foram usadas as seguintes chaves no desenvolvimento do software:
`'brand_raw'` - Para obter o modelo do processador.
`'arch'` - Para obter a arquitetura do processador.
`'bits'` - Para obter o tamanho da palavra do processador.

PLATFORM

O módulo platform é uma biblioteca que tem como objetivo acessar diversas informações do sistema como, hardware, sistema operacional assim como sua versão instalada.

As seguintes funções desse módulo foram utilizadas no software:

`platform.node()` - Retorna uma string com o nome da máquina que é registrada no sistema.
`platform.system()` - Retorna uma string com o nome do sistema operacional da máquina, como 'Linux', 'Darwin', 'Java', 'Windows'.
`platform.version()` - Retorna a versão do sistema operacional.
`platform.machine()` - Retorna a arquitetura da máquina.
`platform.processor()` - Retorna o nome do processador.

PSUTIL

O módulo PSUTIL é uma biblioteca que multiplataforma que tem como objetivo principal recolher informações do sistema, como informações sobre processos em execução e utilização de recursos do sistema como CPU, memória, unidades de armazenamento, rede e sensores.

As seguintes funções do módulo PSUTIL foram utilizadas para coletar as informações do sistema que são exibidas no software.

`psutil.cpu_percent(interval=0.1)` - Mede a porcentagem de utilização da CPU durante um certo intervalo de tempo.

`psutil.virtual_memory()` - Retorna uma tupla com diversas informações da memória principal do sistema. Das informações fornecidas, foram utilizadas apenas a quantidade de total de memória, a quantidade de memória livre e a quantidade de memória utilizada pelo sistema.

`psutil.disk_usage('/')` - Coleta informações referentes a unidade principal de disco. As informações utilizadas foram capacidade total, quantidade livre e quantidade utilizada da unidade de armazenamento.

`psutil.cpu_count(logical=True)` - Retorna à quantidade de unidades de processamento lógicas no sistema.

`psutil.net_if_addrs()` - Retorna diversas informações referentes a placa de rede do sistema. Essa função foi utilizada para coletar o endereço de IP do sistema.

`psutil.cpu_percent(interval=1, percpu=True)` - Mede a porcentagem de utilização de cada núcleo lógico da CPU durante um certo intervalo de tempo.

`psutil.cpu_freq().current.current` - Mede a frequência de utilização da CPU naquele instante.

`psutil.AF_LINK` - Retorna o endereço MAC, da máquina do usuário.

`psutil.net_io_counters()` - Retorna um dicionário contendo todas as interfaces de rede como index, seguido de uma tupla contendo todas as informações de rede desta interface.

`.connections()` - Retorna uma tupla com informações referentes as conexões de rede do processo.

PYGAME

O módulo PYGAME é uma biblioteca projetada para escrever videogames, oferecendo diversas ferramentas para o desenvolvimento completo de jogos e outros programas multimídia na linguagem Python.

Dentre as diversas funções fornecidas pelo PYGAME, as seguintes foram utilizadas para o desenvolvimento do software:

`pygame.display.set_mode()` - Cria uma janela das dimensões que são passadas como parâmetro para a função.

`pygame.display.set_caption()` - Recebe uma string como parâmetro e a utiliza de nome para janela criada na função anterior, pygame.display.set_mode().

`pygame.display.init()` - Inicia a janela com as dimensões e nome que foram passados nas funções anteriores, pygame.display.set_mode() e pygame.display.set_caption().

`pygame.font.init()` - Inicializa o módulo de fontes.

`pygame.font.get_fonts()` - Recebe todas as fontes que estão disponíveis na máquina.

`pygame.font.SysFont(choice, size)` - Cria uma fonte a partir das fontes no sistema.

`pygame.font.Font(None, size)` - Cria uma fonte a partir de um arquivo.

`pygame.surface.Surface()` - Cria superfícies com as dimensões que são passadas como parâmetros.

`pygame.draw.rect()` - Desenha um retângulo com as dimensões e na posição que são passados como parâmetro da função.

`pygame.draw.line()` - Desenha uma linha que se inicia e termina nas posições que são passadas como parâmetro, e tem a espessura também passada como parâmetro.

`pygame.time.Clock()` - Cria um objeto para ajudar a contar o tempo.

`pygame.event.get()` - Fica rastreando por eventos definidos por outras funções.

`pygame.QUIT` - Fica rastreando pelo evento de fechar a janela iniciada pela função pygame.display.init().

`pygame.display.update()` - Atualiza a tela, iniciada pela função pygame.display.init(), a uma frequência definida.

`pygame.display.quit()` - Fecha a tela iniciada anteriormente pela função pygame.display.init().

SOCKET

O módulo Socket fornece acesso à interface de soquete BSD. Está disponível em todos os sistemas Unix modernos, Windows, MacOS e provavelmente em plataformas adicionais. A função socket () retorna um objeto socket cujos métodos implementam as várias chamadas de sistema de socket. Os tipos de parâmetro são de nível um pouco mais alto do que na interface C: como com as operações read() e write() em arquivos Python, a alocação de buffer nas operações de recepção é automática e o comprimento do buffer está implícito nas operações de envio.

As seguintes funções desse módulo foram utilizadas no software:

`socket.AF_INET` - Retorna os endereços IPV4 na máquina do usuário. Com ela é

recebido uma Tupla contendo os diversos endereços utilizados pelo usuário.

`socket.AF_INET6` - Retorna os endereços IPV6 na máquina do usuário. Com ela é recebido uma Tupla contendo os diversos endereços utilizados pelo usuário.

`socket.SOCK_STREAM` - Retorna uma conexão TCP.

`socket.SOCK_DGRAM` - Retorna uma conexão UDP.

`socket.socket()` - Cria um socket para conexão remota, o tipo de IP e de conexão são passados como parâmetro.

`socket.bind()` - Vincula um endereço IP e uma porta a um socket criado anteriormente.

`socket.listen()` - Habilita o socket criado a receber conexão.

`socket.accept()` - Aceita tentativa de conexão no socket.

`socket.send()` - Envia bytes para o cliente ou servidor.

`socket.recv()` - Recebe bytes do cliente ou servidor.

OS

Este módulo fornece uma maneira portátil de usar a funcionalidade dependente do sistema operacional. Ele fornece diversas funções para a manipulação de arquivos, como; leitura ou escrita de arquivo com a função `openT()`, manipulação dos caminhos com o módulo `os.path`, entre outras diversas ferramentas por ele fornecida.

As seguintes funções desse módulo foram utilizadas no software:

`os.path.dirname(path)` - Retorna o nome do diretório inserido em ‘path’.

`os.path.realpath(path)` - Retorna o real caminho do arquivo inserido em ‘path’, eliminando quaisquer links simbólicos encontrados no caminho.

`os.listdir(path)` - Retorna uma lista contendo os nomes das entradas no diretório fornecido por caminho.

`os.path.join(path, *paths)` - Junta um ou mais componentes de caminhos fornecidos de forma inteligente. O valor de retornado é a concatenação do path e quaisquer membros de *paths com exatamente um separador de diretório.

`os.path.isfile(path)` - Retorna True se o objeto contido em path for um arquivo.

`os._exit()` - Encerra todos os processos pendente e em execução.

DATETIME

O módulo DATETIMEfornece classes para manipulação de datas e horas.

A seguinte função desse módulo foi utilizada no software:

`datetime.datetime.fromtimestamp(timestamp)` - Retorna formatado em ano-mês-dia hora:minutos:segundos, o horário fornecido.

TIME

O módulo time fornece várias funções relacionadas a tempo. Essas funções podem ser usdas para cronometrar tempo de processos, tempo de execução de funções e com isso realizar diversos testes de performance do código.

As seguintes funções desse módulo foram utilizadas no software:

`time.clock()` - função removida da versão 3.8 do Python. Não utilizada no Programa.

`time.process_time()` - A função retorna um valor flutuante de tempo em segundos. Retorna o valor da soma do sistema e do tempo de CPU do usuário do processo atual.

`time.perf_counter()` - Retorna um valor flutuante de tempo em segundos. Retorna o valor de um contador de desempenho, ou seja, um relógio com a maior resolução disponível para medir uma curta duração. Esta função Inclui o tempo decorrido `time.sleep()` e abrange todo o sistema. O ponto de referência do valor retornado é indefinido, de forma que apenas a diferença entre os resultados das chamadas consecutivas é válida.

`time.sleep()` - Suspende a execução por um determinado número de segundos. O argumento pode ser um número de ponto flutuante para indicar um tempo de sono mais preciso.

SCHED

O módulo Sched é uma biblioteca padrão do Python, ela implementa um planejador de eventos genérico para executar tarefas em horários específicos. Ele fornece ferramentas semelhantes, como o agendador de tarefas no Windows ou Linux, mas a principal vantagem é que as diferenças entre as plataformas do módulo de Sched do Python podem ser ignoradas.

As seguintes funções desse módulo foram utilizadas no software:

`scheduler.enter()` - Programa um evento para atrasar mais unidades de tempo. Podendo também atribuir diferentes prioridades para os eventos agendados. Além do tempo relativo

`scheduler.run()` - Executa todos os eventos agendados.

THREADING

O módulo Threading é uma biblioteca padrão do Python, ela fornece diversas funções para a criação e manipulação de threads em Python.

As seguintes funções desse módulo foram utilizadas no software:

`threading.Thread()` - Cria uma thread para execução de uma tarefa específica passada como parâmetro.
`.start()` - Inicia a execução da thread criada.
`.join()` - Espere até que o thread termine. Isso bloqueia o encadeamento de chamada até que o encadeamento cujo método join() é chamado termine - normalmente ou por meio de uma exceção não tratada - ou até que ocorra o tempo limite opcional.

SUBPROCESS

O módulo de subprocesso permite gerar novos processos, conectar-se a seus canais de entrada / saída e obter seus códigos de retorno.

A seguinte função desse módulo foi utilizada no software:

`subprocess.call()` - pode ser usado para iniciar um programa. O parâmetro é uma lista da qual o primeiro argumento deve ser o nome do programa.

NMAP

Nmap ("Network Mapper") é um utilitário gratuito e de código aberto (licença) para descoberta de rede e auditoria de segurança. Muitos sistemas e administradores de rede também o consideram útil para tarefas como inventário de rede, gerenciamento de agendas de atualização de serviço e monitoramento de host ou tempo de atividade de serviço. O Nmap usa pacotes IP brutos de novas maneiras para determinar quais hosts estão disponíveis na rede, quais serviços (nome do aplicativo e versão) esses hosts estão oferecendo, quais sistemas operacionais (e versões de SO) eles estão executando, que tipo de filtros de pacotes / firewalls estão em uso e dezenas de outras características.

A seguinte função desse módulo foi utilizada no software:

`nmap.PortScanner()` - Retorna informações a respeito das portas IPs da rede local.

PICKLE

O módulo Pickle implementa protocolos binários para serializar e desserializar uma estrutura de objeto Python.

As seguintes funções desse módulo foram utilizadas no software:

`pickle.dumps()` - Converte o objeto recebido como parâmetro para bytes.
`pickle.loads()` - Converte bytes recebidos como parâmetros, de volta para objetos.

Diferenças entre os Módulos OS e PSUTIL

O módulo PSUTIL é uma biblioteca que tem como objetivo coletar informações a respeito do sistema operacional e, como processos e suas execuções, utilização a utilização dos recursos de hardware da máquina como CPU, memória, unidades de armazenamento, rede e sensores. Já o módulo OS fornece uma maneira de usar a funcionalidade dependente do sistema operacional. Ele fornece diversas funções para a leitura, escrita, manipulação de caminhos, criação e exclusão de arquivos. São módulos com objetivos diferentes, mas cada um atende muito bem a sua proposta.

Glossário

Protocolo de Internet

O protocolo de Internet, ou o IP, do inglês Internet Protocol, é um protocolo necessário para fazer a identificação e a comunicação de máquinas dentro de uma rede. Sem ele, não haveria maneira de saber para onde e/ou de onde as informações de uma rede deveriam ser encaminhadas ou chegaram, se tornando impossível a comunicação. É como você tentar enviar uma encomenda sem os endereços de remetente ou destinatário.

O IP consiste em uma sequência numérica que é utilizado para a identificação de uma máquina na rede. Toda e qualquer máquina que está conectada a uma rede possui um endereço de IP, desde computadores, smartphones, televisores, impressoras e qualquer outro aparelho. E esse número de IP é exclusivo de cada aparelho e toda vez que ele é conectado em uma rede, ele recebe um número de IP.

Atualmente, a maior parte dos endereços de IP são compostos por quatro blocos numéricos, como por exemplo: 192.132.1.31. E existem duas versões do protocolo de IP que são utilizadas. O Protocolo IPv4 define um endereço IP como um número de 32 bits. E o Protocolo IPv6, que surgiu devido ao grande crescimento da Internet, se fazendo necessário uma maior quantidade de endereços de IP. O IPv6 usa 128 bits para o endereçamento de IP.

Arquiteturas de CPU

Atualmente as Unidades de Processamento Central podem ser projetadas utilizando um dos dois tipos de arquitetura que são usadas hoje em dia, a arquiteturas RISC (Reduced Instruction Set Computer) ou arquitetura CISC (Complex Instruction Set Computer).

A arquitetura RISC (Reduced Instruction Set Computer), como o próprio nome já diz, tem como principal objetivo simplificar as instruções de modo que elas possam ser executadas mais rapidamente. Cada instrução executa apenas uma operação, que são todas do mesmo tamanho, tem poucos formatos, e todas as operações aritméticas devem ser executadas entre registradores (dados da memória não podem ser utilizados como operandos). Praticamente todos os conjuntos de instruções (para qualquer arquitetura) lançados desde 1982 têm sido RISC, ou alguma combinação entre RISC e CISC. Membros da família x86 de arquitetura Intel são conhecidos como máquinas CISC, enquanto a Sparc (Sun), Mips (Silicon Graphics), Power (IBM) entre outros, utilizam a arquitetura RISC.

Já a arquitetura CISC (Complex Instruction Set Computer) têm um conjunto de instruções grande, de tamanhos variáveis, com formatos complexos. Muitas dessas instruções são bastante complicadas, executando múltiplas operações quando uma única instrução é dada (por exemplo, é possível realizar um loop complexo usando apenas uma operação assembly). O problema básico com máquinas CISC é que um conjunto pequeno de instruções complexas torna o sistema consideravelmente mais lento. Os projetistas decidiram retornar a uma arquitetura mais simples, delegando ao compilador a responsabilidade de produzir código eficiente com esse novo conjunto de instruções.

Palavra do Processador

Em computação temos os bits como a menor unidade de dados, os bytes, que são um conjunto de 8 bits, como menor unidade da informação e temos a palavra, que é um conjunto de bytes.

Quando falamos em palavra estamos falando de um dado que possui um tamanho/comprimento/largura de bits fixo que aquela arquitetura trabalha melhor. E quando nos referimos a palavra de um processador, em geral, estamos falando do tamanho do registrador do processador. Pelo menos dos registradores principais.

Normalmente a palavra de um processador também determine o tamanho do endereçamento de memória máximo teórico. Se o maior endereço possível tem 32 bits é melhor o processador ter registrador com uma palavra de 32 bits. Arquiteturas mais antigas e algumas muito simples (dispositivos embarcados) podem precisar de mais de um registrador para lidar com os endereços.

Ou seja, quanto maior a palavra de um processador, maior é a capacidade de endereçamento de memória que aquele processador terá.

Núcleos de Processadores

Atualmente todos os processadores possuem núcleos físicos e lógicos, mas nem sempre foi assim. Antigamente os processadores possuem somente núcleos físicos, no entanto, conforme processadores mais potentes foram surgindo e a demanda por mais capacidade de processamento, foram desenvolvidas técnicas de chamadas de Multithreading. Com o Multithreading parte da capacidade de processamento dos processadores físicos eram utilizadas para criar outros processadores lógicos. Aumentando assim a capacidade execução de instruções por segundo da máquina, uma vez que uma vez que o processador estaria utilizando uma maior capacidade dele.

No entanto o Multithreading, por dividir a capacidade de processamento de um núcleo físico em um núcleo lógico, deixa os núcleos físico com uma menor capacidade de processamento que pode ser percebida em operações que alta demanda. Também, para um melhor aproveitamento dos núcleos disponíveis no processador, é necessário que os softwares sejam desenvolvidos considerando a utilização dos núcleos lógicos disponíveis na máquina. Caso contrário pode resultar em alguns núcleos sobrecarregados de operação enquanto outros núcleos estão ociosos. Mas este problema é cada vez mais raro, uma vez que o Multithreading se tornou algo comum nos processadores atuais, e os desenvolvedores vêm se adaptando a eles.

O Problema da Frequência

A explicação mais detalhada sobre por que a consulta do Win32_Processor do WMI (Windows Management Instrumentation) para CurrentClockSpeed parece sempre retornar a frequência máxima em vez da "Velocidade do clock atual"? Na verdade, por que todas as dezenas de contadores WMI / CMI / Perfmon parecem retornar a frequência

"errada"? Se a CPU-Z e o Gerenciador de tarefas puderem obtê-lo, o que temos que fazer para obter a frequência "real"? Para responder a isso, precisamos entender o que CurrentClockSpeed está realmente retornando.

Na documentação WMI para Win32_Processor CurrentClockSpeed temos:

> *Velocidade atual do processador, em MHz. Este valor vem do membro Current Speed da estrutura Processor Information nas informações SMBIOS.*

Com essa informação poderíamos pensar que essa consulta simples deveria nos fornecer a frequência atual. E era exatamente isso que acontecia, e funcionava bem, há doze anos, mas hoje em dia não funciona; porque realmente só funciona para dois casos muito específicos:

1 - Quando você tem um processador que só funciona na velocidade de de fábrica, stock, definida.

2 - Quando um processador móvel é solicitado pelo Windows para funcionar em uma velocidade diferente (por exemplo, mudando para o modo de economia de bateria).

Na inicialização, o Windows obtém as informações do processador e a velocidade do clock atual. A maioria das pessoas está executando seu processador nas configurações recomendadas, sendo assim Current Clock Speed == Max Clock Speed, o que significa que os dois números coincidem o tempo todo. Quando você altera os estados de energia, o Windows altera a frequência e CurrentClockSpeed também.
O que aconteceu há doze anos para tornar o CurrentClockSpeed completamente impreciso e foi uma nova tecnologia, introduzida pela Intel, chamada de Turbo Boost.

O Turbo Boost altera dinamicamente a frequência do processador com base na carga atual do processador dentro dos limites de voltagem, corrente e envelopes térmicos. Quase todos os processadores modernos também têm modos de economia de energia e podem alterar dinamicamente suas frequências, comumente chamado de Turbo Boost, quando a frequência aumenta, e Cool'N'Quiet, quando a frequência diminui.

O grande problema é que toda essa variação na frequência do processador é feita automaticamente, sem que o Windows saiba disso. Como o Windows não sabe sobre isso, o valor CurrentClockSpeed pode ser completamente impreciso na maioria das vezes.

Este é um problema conhecido pela Microsoft, e é por isso quando se abre o Monitor de Desempenho e olha a descrição em Desempenho do Processador / Freqüência do Processador, tem a seguinte mensagem:

> *A frequência do processador é a frequência do processador atual em megahertz. Alguns processadores são capazes de regular sua frequência fora do controle do Windows. A frequência do processador não refletirá com precisão a frequência real*

do processador nesses sistemas. Em vez disso, use Informações do processador \% Desempenho do processador.

Diferença entre Clock da CPU e Tempo Real

O Clock é a frequência, medida em Hz, do número de instruções que o processador é capaz de executar em 1 segundo. Ou seja, quanto maior a frequência (o clock) de um processador, menor será o tempo em que um processo estará em sendo executado pela CPU. Mas, por outro lado, quanto mais “pesada”, complexa, for essa tarefa, maior será o tempo de execução dessa tarefa pela CPU. Já o Tempo Real é o tempo desde o início da criação de um processo até o processo ser concluído incluindo os tempos de espera.

Bibliografia

ITS INSTITUTE FOR TELECOMMUNICATIONMOBILON NETWORKS Disponível em: <https://www.its.bldrdoc.gov/>. Acesso em: 26 de Setembro de 2020.

PSUTIL Documentation; Disponível em: <https://psutil.readthedocs.io/en/latest//>. Acesso em: 26 de Setembro de 2020.

PLATFORM — Documentation; Disponível em: <https://docs.python.org/3/library/platform.html>. Acesso em: 26 de Setembro de 2020.

PYGAME Documentation; Disponível em: <https://www.pygame.org/docs/>. Acesso em: 26 de Setembro de 2020.

SOCKET — Documentation; Disponível em: <https://docs.python.org/3/library/socket.html>. Acesso em: 26 de Setembro de 2020.

OS — Documentation; Disponível em: <https://docs.python.org/3/library/os.html>. Acesso em: 26 de Setembro de 2020.

SCHED — Documentation; Disponível em: <https://docs.python.org/3/library/sched.html>. Acesso em: 26 de Setembro de 2020.

TIME — Documentation; Disponível em: <https://docs.python.org/3/library/time.html>. Acesso em: 26 de Setembro de 2020.

THREADING— Documentation; Disponível em: <https://docs.python.org/3/library/threading.html>. Acesso em: 26 de Setembro de 2020.

PyGame Tutorial: Fonts and Text Disponível em: <https://nerdparadise.com/programming/pygame/part5>. Acesso em: 26 de Setembro de 2020.

StakOverflow, HAL9256; Disponível em: <https://stackoverflow.com/questions/61802420/unable-to-get-current-cpu-frequency-in-powershell-or-python>. Acesso em: 26 de Setembro de 2020.

PATTERSON, David A.; HENNESSY, John L. Organização e Projeto de Computadores. Tradução da 3a edição. Editora Campus, 2005.

STALLINGS, William. Arquitetura e Organização de Computadores. Tradução da 8a edição. Editora Prentice Hall Brasil, 2002.

TANENBAUM, Andrew S. Organização Estruturada de Computadores. Tradução da 6a edição. Editora Prentice Hall Brasil, 2013.

MONTEIRO, Mário A. Introdução à Organização de Computadores. Tradução da 5a edição. Editora LTC, 2007.

Anexo I

Códigos

Código da Versão 1.0

```
import psutil
import platform
import pygame

### Windows Config. ###

width_window = 800
height_window = 600

window = pygame.display.set_mode((width_window, height_window))
pygame.display.set_caption("PC Resources Monitor")
pygame.display.init()

### Fonts ###

pygame.font.init()

def make_font(fonts, size):
    available = pygame.font.get_fonts()
    # get_fonts() returns a list of lowercase spaceless font names
    choices = map(lambda x:x.lower().replace(' ', ''), fonts)
    for choice in choices:
        if choice in available:
            return pygame.font.SysFont(choice, size)
    return pygame.font.Font(None, size)

_cached_fonts = {}

def get_font(font_preferences, size):
    global _cached_fonts
    key = str(font_preferences) + '|' + str(size)
    font = _cached_fonts.get(key, None)
    if font == None:
        font = make_font(font_preferences, size)
        _cached_fonts[key] = font
    return font

font_preferences = ["Consolas","Rockwell", "Calibri", "Arial", ]

font_1 = get_font(font_preferences, 30)
font_2 = get_font(font_preferences, 20)
font_pc = get_font(font_preferences, 12)

### Colors ###

red = (200,0,0)
green = (255,0,0)
blue = (0,0,255)
black = (0,0,0)
white = (255, 255, 255)
grey = (150, 150, 150)

light_blue = (0,200,255)
grey_blue = (0, 55, 130)
dark_blue = (0, 0, 50)
sand = (255, 230, 185)
sand2 = (225, 190, 145)
dark_sand = (165, 145, 110)
```

```
background = (40, 40, 55)

### Surfaces ###

s_cpu = pygame.surface.Surface((width_window, (height_window)/4))
s_mem = pygame.surface.Surface((width_window, (height_window)/4))
s_hd = pygame.surface.Surface((width_window, (height_window)/4))
s_pc = pygame.surface.Surface((width_window, (height_window)/4))

### CPU Data ###

def cpu():
    cpu = psutil.cpu_percent(interval=0.1)

    width_bar1 = width_window - 2*20
    s_cpu.fill(background)

    pygame.draw.rect(s_cpu, black, (25, 70, width_bar1, 70))
    pygame.draw.rect(s_cpu, grey_blue, (20, 65, width_bar1, 70))

    width_bar2 = ((width_bar1*cpu) / 100)

    pygame.draw.rect(s_cpu, black, (25, 70, width_bar2, 70))
    pygame.draw.rect(s_cpu, red, (20, 65, width_bar2, 70))

    pygame.draw.line(s_cpu, black, (10,54), (790, 54), 6)

    bar_text = "CPU Usage"
    bar_text_inside1 = str(cpu) + "%"
    bar_text_inside2 = str((100 - cpu)) + "%"

    text = font_1.render(bar_text, 10, grey)
    text_inside1 = font_2.render(bar_text_inside1, 10, sand2)
    text_inside2 = font_2.render(bar_text_inside2, 10, dark_sand)

    s_cpu.blit(text, (20, 10))
    s_cpu.blit(text_inside1, (35, 110))
    s_cpu.blit(text_inside2, (700, 110))

    window.blit(s_cpu, (0, 0))

### Memory Data ###

def memory():
    mem = psutil.virtual_memory()

    memTotalGigas = round((mem.total / (1024*1024*1024)), 2)
    memUsedGigas = round((mem.used / (1024*1024*1024)), 2)
    memFreeGigas = round((mem.free / (1024*1024*1024)), 2)

    width_bar1 = width_window - 2*20
    s_mem.fill(background)

    pygame.draw.rect(s_mem, black, (25, 70, width_bar1, 70))
    pygame.draw.rect(s_mem, grey_blue, (20, 65, width_bar1, 70))

    width_bar2 = ((width_bar1*mem.percent) / 100)

    pygame.draw.rect(s_mem, black, (25, 70, width_bar2, 70))
    pygame.draw.rect(s_mem, red, (20, 65, width_bar2, 70))
```

```
    pygame.draw.line(s_mem, black, (10,45), (790, 45), 6)

    bar_text_1 = "Memory Usage"
    bar_text_2 = "(Total: " + str(memTotalGigas) + " GB)"
    bar_text_inside1 = str(memUsedGigas) + " Used"
    bar_text_inside2 = str(memFreeGigas) + " Free"

    text_1 = font_1.render(bar_text_1, 10, grey)
    text_2 = font_1.render(bar_text_2, 10, grey)
    text_inside1 = font_2.render(bar_text_inside1, 10, sand2)
    text_inside2 = font_2.render(bar_text_inside2, 10, dark_sand)


    s_mem.blit(text_1, (20, 10))
    s_mem.blit(text_2, (480, 10))
    s_mem.blit(text_inside1, (35, 110))
    s_mem.blit(text_inside2, (655, 110))

    window.blit(s_mem, (0, ((height_window)/4)))

### HD Data ###

def hd():
    hd = psutil.disk_usage('/')

    hdTotalGigas = round((hd.total / (1024*1024*1024)), 2)
    hdUsedGigas = round((hd.used / (1024*1024*1024)), 2)
    hdFreeGigas = round((hd.free / (1024*1024*1024)), 2)

    width_bar1 = width_window - 2*20
    s_hd.fill(background)

    pygame.draw.rect(s_hd, black, (25, 70, width_bar1, 70))
    pygame.draw.rect(s_hd, grey_blue, (20, 65, width_bar1, 70))

    width_bar2 = ((width_bar1*hd.percent) / 100)

    pygame.draw.rect(s_hd, black, (25, 70, width_bar2, 70))
    pygame.draw.rect(s_hd, red, (20, 65, width_bar2, 70))

    pygame.draw.line(s_hd, black, (10,45), (790, 45), 6)

    bar_text_1 = "Primary Storage Usage"
    bar_text_2 = "(Total: " + str(hdTotalGigas) + " GB)"
    bar_text_inside1 = str(hdUsedGigas) + " Used"
    bar_text_inside2 = str(hdFreeGigas) + " Free"

    text_1 = font_1.render(bar_text_1, 10, grey)
    text_2 = font_1.render(bar_text_2, 10, grey)
    text_inside1 = font_2.render(bar_text_inside1, 10, sand2)
    text_inside2 = font_2.render(bar_text_inside2, 10, dark_sand)

    s_hd.blit(text_1, (20, 10))
    s_hd.blit(text_2, (475, 10))
    s_hd.blit(text_inside1, (35, 110))
    s_hd.blit(text_inside2, (640, 110))

    window.blit(s_hd, (0, ((((height_window)/4)*2))))

### PC Data ###
```

```
def pc():

    pc_name = platform.node()
    os = platform.system()
    osVersion = platform.version()
    arc = platform.machine()
    processor = platform.processor()
    cores = psutil.cpu_count(logical=True)

    ### IPs ###
    import socket

    def get_ip_addresses(family):
        for interface, snics in psutil.net_if_addrs().items():
            for snic in snics:
                if snic.family == family:
                    yield (interface, snic.address)

    ipv4s = list(get_ip_addresses(socket.AF_INET))
    ipv6s = list(get_ip_addresses(socket.AF_INET6))

    ip = ipv4s[0][1]

    s_pc.fill(background)

    pygame.draw.line(s_pc, black, (10,45), (790, 45), 6)

    text_0 = "PC Information"
    text_1 = "PC Name: " + str(pc_name)
    text_2 = "Operating System: " + str(os) + " " + str(osVersion)
    text_3 = "CPU Architecture: " + str(arc)
    text_4 = "Processor Name: " + str(processor)
    text_5 = "Cores: " + str(cores)
    text_6 = "Local IP Address: " + str(ip)

    font_text_0 = font_1.render(text_0, 10, grey)
    font_text_1 = font_pc.render(text_1, 10, dark_sand)
    font_text_2 = font_pc.render(text_2, 10, dark_sand)
    font_text_3 = font_pc.render(text_3, 10, dark_sand)
    font_text_4 = font_pc.render(text_4, 10, dark_sand)
    font_text_5 = font_pc.render(text_5, 10, dark_sand)
    font_text_6 = font_pc.render(text_6, 10, dark_sand)


    s_pc.blit(font_text_0, (20, 10))
    s_pc.blit(font_text_1, (20, 55))
    s_pc.blit(font_text_2, (20, 70))
    s_pc.blit(font_text_3, (20, 85))
    s_pc.blit(font_text_4, (20, 100))
    s_pc.blit(font_text_5, (20, 115))
    s_pc.blit(font_text_6, (20, 130))

    window.blit(s_pc, (0, ((((height_window)/4)*3))))

### Display ###

clock = pygame.time.Clock()
cont = 60
close = False

while not close:
```

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            close = True

    if cont == 60:
        cpu()
        hd()
        memory()
        pc()

        cont = 0

    pygame.display.update()
    clock.tick(60)

pygame.display.quit()
```

Código da Versão 2.0

```
import psutil
import cpuinfo
import platform
import pygame

### Windows Config. ###

width_window = 860
height_window = 600

window = pygame.display.set_mode((width_window, height_window))
pygame.display.set_caption("PC Resources Monitor")
pygame.display.init()

### Fonts ###

pygame.font.init()

def make_font(fonts, size):
    available = pygame.font.get_fonts()
    # get_fonts() returns a list of lowercase spaceless font names
    choices = map(lambda x:x.lower().replace(' ', ''), fonts)
    for choice in choices:
        if choice in available:
            return pygame.font.SysFont(choice, size)
    return pygame.font.Font(None, size)

_cached_fonts = {}

def get_font(font_preferences, size):
    global _cached_fonts
    key = str(font_preferences) + '|' + str(size)
    font = _cached_fonts.get(key, None)
    if font == None:
        font = make_font(font_preferences, size)
        _cached_fonts[key] = font
    return font

font_preferences = ["Consolas","Rockwell", "Calibri", "Arial", ]

font_1 = get_font(font_preferences, 30)
font_2 = get_font(font_preferences, 20)
font_pc = get_font(font_preferences, 12)

### Colors ###

red = (200,0,0)
green = (255,0,0)
blue = (0,0,255)
black = (0,0,0)
white = (255, 255, 255)
grey = (150, 150, 150)

light_blue = (0,200,255)
grey_blue1 = (156, 181, 201)
grey_blue2 = (0, 55, 130)
dark_blue = (0, 0, 50)
sand = (255, 230, 185)
```

```
sand2 = (225, 190, 145)
dark_sand = (165, 145, 110)

background = (40, 40, 55)

### Surfaces ###

s_cpu = pygame.surface.Surface((800, (height_window)/4))
s_mem = pygame.surface.Surface((800, (height_window)/4))
s_hd = pygame.surface.Surface((800, (height_window)/4))
s_pc_1 = pygame.surface.Surface((800, (height_window)/4))
s_pc_2 = pygame.surface.Surface((800, (height_window)/4))
s_cpu_graph_back = pygame.surface.Surface((800, ((height_window)/4)))
s_cpu_graph_1 = pygame.surface.Surface((800, ((height_window)/4)-58))
s_cpu_graph_2 = pygame.surface.Surface((800, ((height_window)/4)*3))

s_menu = pygame.surface.Surface((60, height_window))
s_display = pygame.surface.Surface((60, 300))

### Menu ###

s_menu.fill(background)

pygame.draw.circle(s_menu, black, (25,25), 15)
botton_menu_1 = pygame.draw.circle(s_menu, grey_blue1, (25,25), 11)

pygame.draw.circle(s_menu, black, (25,475), 15)
botton_menu_2 = pygame.draw.circle(s_menu, grey_blue1, (25,475), 11)

pygame.draw.line(s_menu, black, (50,0), (50, height_window), 10)

window.blit(s_menu, (0, 0))


def display_menu(display_surface):

    s_display.fill(background)

    pygame.draw.line(s_display, black, (50,0), (50, height_window), 10)

    if display_surface == 'home':
        #botton home
        pygame.draw.circle(s_display, grey, (25,110), 10)
        botton_display_0 = pygame.draw.circle(s_display, white, (25,110), 7)

        #botton cpu_inf_1
        pygame.draw.circle(s_display, black, (25,140), 8)
        botton_display_1 = pygame.draw.circle(s_display, grey_blue1, (25,140), 5)

        #botton cpu_graph_1
        pygame.draw.circle(s_display, black, (25,170), 8)
        botton_display_2 = pygame.draw.circle(s_display, grey_blue1, (25,170), 5)

        #botton cpu_graph_2
        pygame.draw.circle(s_display, black, (25,200), 8)
        botton_display_3 = pygame.draw.circle(s_display, grey_blue1, (25,200), 5)

    elif display_surface == 'cpu_inf_1':
        #botton home
        pygame.draw.circle(s_display, black, (25,110), 8)
        botton_display_0 = pygame.draw.circle(s_display, grey_blue1, (25,110), 5)
```

```
        #botton cpu_inf_1
        pygame.draw.circle(s_display, grey, (25,140), 10)
        botton_display_1 = pygame.draw.circle(s_display, white, (25,140), 7)

        #botton cpu_graph_1
        pygame.draw.circle(s_display, black, (25,170), 8)
        botton_display_2 = pygame.draw.circle(s_display, grey_blue1, (25,170), 5)

        #botton cpu_graph_2
        pygame.draw.circle(s_display, black, (25,200), 8)
        botton_display_3 = pygame.draw.circle(s_display, grey_blue1, (25,200), 5)

    elif display_surface == 'cpu_graph_1':
        #botton home
        pygame.draw.circle(s_display, black, (25,110), 8)
        botton_display_0 = pygame.draw.circle(s_display, grey_blue1, (25,110), 5)

        #botton cpu_inf_1
        pygame.draw.circle(s_display, black, (25,140), 8)
        botton_display_1 = pygame.draw.circle(s_display, grey_blue1, (25,140), 5)

        #botton cpu_graph_1
        pygame.draw.circle(s_display, grey, (25,170), 10)
        botton_display_2 = pygame.draw.circle(s_display, white, (25,170), 7)

        #botton cpu_graph_2
        pygame.draw.circle(s_display, black, (25,200), 8)
        botton_display_3 = pygame.draw.circle(s_display, grey_blue1, (25,200), 5)

    elif display_surface == 'cpu_graph_2':
        #botton home
        pygame.draw.circle(s_display, black, (25,110), 8)
        botton_display_0 = pygame.draw.circle(s_display, grey_blue1, (25,110), 5)

        #botton cpu_inf_1
        pygame.draw.circle(s_display, black, (25,140), 8)
        botton_display_1 = pygame.draw.circle(s_display, grey_blue1, (25,140), 5)

        #botton cpu_graph_1
        pygame.draw.circle(s_display, black, (25,170), 8)
        botton_display_2 = pygame.draw.circle(s_display, grey_blue1, (25,170), 5)

        #botton cpu_graph_2
        pygame.draw.circle(s_display, grey, (25,200), 10)
        botton_display_3 = pygame.draw.circle(s_display, white, (25,200), 7)

    window.blit(s_display, (0, 120))

### Data ###

# CPU
cpu = int(psutil.cpu_percent(interval=0.1))
cores_percent = psutil.cpu_percent(interval=1, percpu=True)

# Mem
mem = psutil.virtual_memory()
memTotalGigas = round((mem.total / (1024*1024*1024)), 2)
memUsedGigas = round((mem.used / (1024*1024*1024)), 2)
memFreeGigas = round((mem.free / (1024*1024*1024)), 2)
```

```
# HD
hd = psutil.disk_usage('/')
hdTotalGigas = round((hd.total / (1024*1024*1024)), 2)
hdUsedGigas = round((hd.used / (1024*1024*1024)), 2)
hdFreeGigas = round((hd.free / (1024*1024*1024)), 2)

# PC Inf 1
pc_name = platform.node()
os = platform.system()
osVersion = platform.version()
arc = platform.machine()
processor = platform.processor()
cores = psutil.cpu_count(logical=True)

# PC Inf 2
inf = cpuinfo.get_cpu_info()
cpu_name = inf['brand_raw']
cpu_arch = inf['arch']
cpu_bits = inf['bits']
cpu_hz = round(psutil.cpu_freq().current, 2)
p_core = psutil.cpu_count(logical=False)
l_cores = psutil.cpu_count()

# IPs
import socket

def get_ip_addresses(family):
    for interface, snics in psutil.net_if_addrs().items():
        for snic in snics:
            if snic.family == family:
                yield (interface, snic.address)

ipv4s = list(get_ip_addresses(socket.AF_INET))
ipv6s = list(get_ip_addresses(socket.AF_INET6))

ip = ipv4s[0][1]

### CPU Graphic 0 Surface ###

def cpu_graph_0():

    width_bar1 = 800 - 2*20
    s_cpu.fill(background)

    pygame.draw.rect(s_cpu, black, (25, 70, width_bar1, 70))
    pygame.draw.rect(s_cpu, grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = ((width_bar1*cpu) / 100)

    pygame.draw.rect(s_cpu, black, (25, 70, width_bar2, 70))
    pygame.draw.rect(s_cpu, red, (20, 65, width_bar2, 70))

    pygame.draw.line(s_cpu, black, (10,54), (790, 54), 6)

    bar_text = "CPU Usage"
    bar_text_inside1 = str(cpu) + "%"
    bar_text_inside2 = str((100 - cpu)) + "%"

    text = font_1.render(bar_text, 10, grey)
    text_inside1 = font_2.render(bar_text_inside1, 10, sand2)
    text_inside2 = font_2.render(bar_text_inside2, 10, dark_sand)
```

```
    s_cpu.blit(text, (20, 10))
    s_cpu.blit(text_inside1, (35, 110))
    s_cpu.blit(text_inside2, (700, 110))

    window.blit(s_cpu, (60, 0))

### CPU Graphic 1 Surface ###

def cpu_graph_1():

    s_cpu_graph_back.fill(background)
    s_cpu_graph_1.fill(background)

    cores = len(cores_percent)

    x = y = shift = 10

    height = s_cpu_graph_1.get_height() - 2*y
    width = ( s_cpu_graph_1.get_width() - 2*y - (cores + 1)*shift ) / cores

    d = x + shift

    for i in cores_percent:
        pygame.draw.rect(s_cpu_graph_1, black, (d+5, y+5, width, height))
        pygame.draw.rect(s_cpu_graph_1, red, (d, y, width, height))
        pygame.draw.rect(s_cpu_graph_1, grey_blue2, (d, y, width, ((1-i/100)*height)))

        d = d + width + shift

    pygame.draw.line(s_cpu_graph_back, black, (10,54), (790, 54), 6)

    bar_text = "CPU Usage Per Logical Processors"
    text = font_1.render(bar_text, 10, grey)
    s_cpu_graph_back.blit(text, (20, 10))

    window.blit(s_cpu_graph_back, (60, 0))
    window.blit(s_cpu_graph_1, (60, 58))


### CPU Graphic 2 Surface ###

def cpu_graph_2():

    s_cpu_graph_2.fill(background)

    cores = len(cores_percent)

    x = y = shift = 10

    height = s_cpu_graph_2.get_height() - 2*y
    width = ( s_cpu_graph_2.get_width() - 2*y - (cores + 1)*shift ) / cores

    d = x + shift

    for i in cores_percent:
        pygame.draw.rect(s_cpu_graph_2, black, (d+5, y+5, width, height))
        pygame.draw.rect(s_cpu_graph_2, red, (d, y, width, height))
        pygame.draw.rect(s_cpu_graph_2, grey_blue2, (d, y, width, ((1-i/100)*height)))

        d = d + width + shift
```

```
    bar_text = "CPU Usage Per Logical Processors"
    text = font_1.render(bar_text, 10, grey)

    window.blit(s_cpu_graph_2, (60, (height_window)/4))

### CPU Inf Surface ###

def cpu_inf(pos_x, pos_y):

    s_pc_2.fill(background)

    pygame.draw.line(s_pc_2, black, (10,45), (790, 45), 6)

    text_0 = "CPU Information"
    text_1 = "CPU Name: " + str(cpu_name)
    text_2 = "Architecture: " + str(cpu_arch)
    text_3 = "Word Length: " + str(cpu_bits)
    text_4 = "Current Frequency: " + str(cpu_hz)
    text_5 = "Physical Cores: " + str(p_core)
    text_6 = "Logical Cores: " + str(l_cores)

    font_text_0 = font_1.render(text_0, 10, grey)
    font_text_1 = font_pc.render(text_1, 10, dark_sand)
    font_text_2 = font_pc.render(text_2, 10, dark_sand)
    font_text_3 = font_pc.render(text_3, 10, dark_sand)
    font_text_4 = font_pc.render(text_4, 10, dark_sand)
    font_text_5 = font_pc.render(text_5, 10, dark_sand)
    font_text_6 = font_pc.render(text_6, 10, dark_sand)

    s_pc_2.blit(font_text_0, (20, 10))
    s_pc_2.blit(font_text_1, (20, 55))
    s_pc_2.blit(font_text_2, (20, 70))
    s_pc_2.blit(font_text_3, (20, 85))
    s_pc_2.blit(font_text_4, (20, 100))
    s_pc_2.blit(font_text_5, (20, 115))
    s_pc_2.blit(font_text_6, (20, 130))

    #window.blit(s_pc_2, (60, (((height_window)/4)*3)))
    window.blit(s_pc_2, (pos_x, pos_y))

### Memory Graphic 0 Surface ###

def mem_graph_0():

    width_bar1 = 800 - 2*20
    s_mem.fill(background)

    pygame.draw.rect(s_mem, black, (25, 70, width_bar1, 70))
    pygame.draw.rect(s_mem, grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = ((width_bar1*mem.percent) / 100)

    pygame.draw.rect(s_mem, black, (25, 70, width_bar2, 70))
    pygame.draw.rect(s_mem, red, (20, 65, width_bar2, 70))

    pygame.draw.line(s_mem, black, (10,45), (790, 45), 6)

    bar_text_1 = "Memory Usage"
    bar_text_2 = "(Total: " + str(memTotalGigas) + " GB)"
    bar_text_inside1 = str(memUsedGigas) + " Used"
```

```
    bar_text_inside2 = str(memFreeGigas) + " Free"

    text_1 = font_1.render(bar_text_1, 10, grey)
    text_2 = font_1.render(bar_text_2, 10, grey)
    text_inside1 = font_2.render(bar_text_inside1, 10, sand2)
    text_inside2 = font_2.render(bar_text_inside2, 10, dark_sand)


    s_mem.blit(text_1, (20, 10))
    s_mem.blit(text_2, (480, 10))
    s_mem.blit(text_inside1, (35, 110))
    s_mem.blit(text_inside2, (655, 110))

    window.blit(s_mem, (60, ((height_window)/4)))

### HD Graphic 0 Surface ###

def hd_graph_0():

    width_bar1 = 800 - 2*20
    s_hd.fill(background)

    pygame.draw.rect(s_hd, black, (25, 70, width_bar1, 70))
    pygame.draw.rect(s_hd, grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = ((width_bar1*hd.percent) / 100)

    pygame.draw.rect(s_hd, black, (25, 70, width_bar2, 70))
    pygame.draw.rect(s_hd, red, (20, 65, width_bar2, 70))

    pygame.draw.line(s_hd, black, (10,45), (790, 45), 6)

    bar_text_1 = "Primary Storage Usage"
    bar_text_2 = "(Total: " + str(hdTotalGigas) + " GB)"
    bar_text_inside1 = str(hdUsedGigas) + " Used"
    bar_text_inside2 = str(hdFreeGigas) + " Free"

    text_1 = font_1.render(bar_text_1, 10, grey)
    text_2 = font_1.render(bar_text_2, 10, grey)
    text_inside1 = font_2.render(bar_text_inside1, 10, sand2)
    text_inside2 = font_2.render(bar_text_inside2, 10, dark_sand)

    s_hd.blit(text_1, (20, 10))
    s_hd.blit(text_2, (475, 10))
    s_hd.blit(text_inside1, (35, 110))
    s_hd.blit(text_inside2, (640, 110))

    window.blit(s_hd, (60, ((((height_window)/4)*2))))

### PC Inf Surface ###

def pc_1():

    s_pc_1.fill(background)

    pygame.draw.line(s_pc_1, black, (10,45), (790, 45), 6)

    text_0 = "PC Information"
    text_1 = "PC Name: " + str(pc_name)
    text_2 = "Operating System: " + str(os) + " " + str(osVersion)
    text_3 = "CPU Architecture Name: " + str(arc)
```

```
    text_4 = "Processor Name: " + str(processor)
    text_5 = "Cores: " + str(cores)
    text_6 = "Local IP Address: " + str(ip)

    font_text_0 = font_1.render(text_0, 10, grey)
    font_text_1 = font_pc.render(text_1, 10, dark_sand)
    font_text_2 = font_pc.render(text_2, 10, dark_sand)
    font_text_3 = font_pc.render(text_3, 10, dark_sand)
    font_text_4 = font_pc.render(text_4, 10, dark_sand)
    font_text_5 = font_pc.render(text_5, 10, dark_sand)
    font_text_6 = font_pc.render(text_6, 10, dark_sand)


    s_pc_1.blit(font_text_0, (20, 10))
    s_pc_1.blit(font_text_1, (20, 55))
    s_pc_1.blit(font_text_2, (20, 70))
    s_pc_1.blit(font_text_3, (20, 85))
    s_pc_1.blit(font_text_4, (20, 100))
    s_pc_1.blit(font_text_5, (20, 115))
    s_pc_1.blit(font_text_6, (20, 130))

    window.blit(s_pc_1, (60, ((((height_window)/4)*3))))

### Display ###

clock = pygame.time.Clock()
cont = 60
close = False

display_surface = 'home'
botton_1 = 'cpu_graph_0'
botton_2 = 'pc_1'

while not close:

    for event in pygame.event.get():

        ### Keyboard ###

        if event.type == pygame.KEYDOWN:

            if ((event.key == pygame.K_RIGHT) and (display_surface == 'home')) or ((event.key ==
pygame.K_LEFT) and (display_surface == 'cpu_graph_1')):

                cpu_graph_0()
                mem_graph_0()
                hd_graph_0()
                cpu_inf(60, (((height_window)/4)*3))

                display_surface = 'cpu_inf_1'
                botton_1 = 'cpu_graph_0'
                botton_2 = 'cpu_inf'
                display_menu(display_surface)

            elif ((event.key == pygame.K_RIGHT) and (display_surface == 'cpu_inf_1')) or ((event.key ==
pygame.K_LEFT) and (display_surface == 'cpu_graph_2')):

                cpu_graph_1()
                mem_graph_0()
                hd_graph_0()
                cpu_inf(60, (((height_window)/4)*3))
```

```
                display_surface = 'cpu_graph_1'
                botton_1 = 'cpu_graph_1'
                botton_2 = 'cpu_inf'
                display_menu(display_surface)

            elif ((event.key == pygame.K_RIGHT) and (display_surface == 'cpu_graph_1')) or ((event.key ==
pygame.K_LEFT) and (display_surface == 'home')):

                cpu_graph_0()
                cpu_graph_2()

                display_surface = 'cpu_graph_2'
                botton_1 = 'cpu_graph_0'
                botton_2 = 'disable'
                display_menu(display_surface)

            elif ((event.key == pygame.K_RIGHT) and (display_surface == 'cpu_graph_2')) or ((event.key ==
pygame.K_LEFT) and (display_surface == 'cpu_inf_1')) or ((event.key == pygame.K_SPACE)):

                cpu_graph_0()
                mem_graph_0()
                hd_graph_0()
                pc_1()

                display_surface = 'home'
                botton_1 = 'cpu_graph_0'
                botton_2 = 'pc_1'
                display_menu(display_surface)

        ### Mouse ###

        if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:

            pos = pygame.mouse.get_pos()

            if botton_menu_1.collidepoint(pos):

                if botton_1 == 'cpu_graph_1':
                    cpu_graph_0()
                    botton_1 = 'cpu_graph_0'

                elif (botton_1 == 'cpu_graph_0') and (display_surface == 'cpu_graph_2'):
                    cpu_inf(60, 0)
                    botton_1 = 'cpu_inf'

                elif (botton_1 == 'cpu_inf') and (display_surface == 'cpu_graph_2'):
                    cpu_graph_0()
                    botton_1 = 'cpu_graph_0'

                elif (botton_1 == 'cpu_graph_0') and (display_surface != 'cpu_graph_2'):
                    cpu_graph_1()
                    botton_1 = 'cpu_graph_1'

            if botton_menu_2.collidepoint(pos):

                if botton_2 == 'pc_1':
                    cpu_inf(60, (((height_window)/4)*3))
                    botton_2 = 'cpu_inf'

                elif botton_2 == 'cpu_inf':
```

```
                    pc_1()
                    botton_2 = 'pc_1'

        if event.type == pygame.QUIT:
            close = True

    if cont == 60:
        cpu_graph_0()
        mem_graph_0()
        hd_graph_0()
        pc_1()

        display_menu(display_surface)

        cont = 0

    pygame.display.update()
    clock.tick(60)

pygame.display.quit()
```

Código da Versão 3.0

Arquivo: main.py

```
import pygame
import data_and_funcs
import display_interface

def main():
    pygame.display.set_caption("PC Resources Monitor")
    pygame.display.init()

    ### Display ###
    clock = pygame.time.Clock()
    cont = 0
    close = False
    files_first_page = True
    process_first_page = True
    while not close:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                close = True
            ### Keyboard ###
            if event.type == pygame.KEYDOWN:
                if ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'HOME')) or ((event.key ==
pygame.K_LEFT) and (display_interface.menu == 'FILES')):
                    display_interface.menu = 'CPU'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'CPU')) or ((event.key
== pygame.K_LEFT) and (display_interface.menu == 'PROCESS')):
                    display_interface.menu = 'FILES'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'FILES')) or ((event.key
== pygame.K_LEFT) and (display_interface.menu == 'HOME')):
                    display_interface.menu = 'PROCESS'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'PROCESS')) or
((event.key == pygame.K_LEFT) and (display_interface.menu == 'CPU')) or ((event.key == pygame.K_SPACE)):
                    display_interface.menu = 'HOME'
            ### Mouse ###
            if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                pos = pygame.mouse.get_pos()
                if display_interface.botton_display_0.collidepoint(((pos[0] - 200), (pos[1] -
(display_interface.WINDOW_HEIGHT + 5)))):
                    display_interface.menu = 'HOME'
                elif display_interface.botton_display_1.collidepoint(((pos[0] - 200), (pos[1] -
(display_interface.WINDOW_HEIGHT + 5)))):
                    display_interface.menu = 'CPU'
                elif display_interface.botton_display_2.collidepoint(((pos[0] - 200), (pos[1] -
(display_interface.WINDOW_HEIGHT + 5)))):
                    display_interface.menu = 'FILES'
                elif display_interface.botton_display_3.collidepoint(((pos[0] - 200), (pos[1] -
(display_interface.WINDOW_HEIGHT + 5)))):
                    display_interface.menu = 'PROCESS'
        cont += 1
        if cont == 5:
            display_interface.menu_buttons(display_interface.menu)
            if display_interface.menu == 'HOME':
                data_and_funcs.cpu_graph_0()
                data_and_funcs.mem_graph_0()
                data_and_funcs.hd_graph_0()
                data_and_funcs.pc_1()
            elif display_interface.menu == 'CPU':
                data_and_funcs.cpu_graph_0()
                data_and_funcs.cpu_graph_1()
```

```
                data_and_funcs.cpu_inf()
            elif display_interface.menu =='FILES':

                files = data_and_funcs.dir_files()
                if files_first_page:
                    files_left = data_and_funcs.files_page(files)
                if data_and_funcs.files_next_page(files_left):
                    if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                        pos = pygame.mouse.get_pos()
                        if (pos[0] - 205) >= (520) and (pos[0] - 205) <= (540) and (pos[1] - 5) >= (30) and
(pos[1] - 5) <= (50):
                            files_left = data_and_funcs.files_page(files_left)
                            files_first_page = False
            elif display_interface.menu == 'PROCESS':
                process = data_and_funcs.running_processes()
                if process_first_page:
                    process_left = data_and_funcs.processes_page(process)
                if data_and_funcs.process_next_page(process_left):
                    if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                        pos = pygame.mouse.get_pos()
                        if (pos[0] - 205) >= (520) and (pos[0] - 205) <= (540) and (pos[1] - 5) >= (30) and
(pos[1] - 5) <= (50):
                            process_left = data_and_funcs.processes_page(process_left)
                            process_first_page = False
            cont = 0
        pygame.display.update()
        clock.tick(60)
    pygame.display.quit()
if __name__ == "__main__":
    main()
```

Arquivo: display_interface.py

```
import pygame
import main


### Windows Config. ###
WINDOW_WIDTH = 800
WINDOW_HEIGHT = 600
MENU_BORDER_WIDTH = WINDOW_WIDTH
MENU_BORDER_HEIGHT = 40
MENU_WIDTH = (WINDOW_WIDTH - 10)
MENU_HEIGHT = (MENU_BORDER_HEIGHT - 10)

window = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT + MENU_BORDER_HEIGHT))

### Surfaces ###
s_menu = pygame.surface.Surface((MENU_WIDTH, MENU_HEIGHT))
s_menu_border = pygame.surface.Surface((MENU_BORDER_WIDTH, MENU_BORDER_HEIGHT))
s_display = pygame.surface.Surface(((MENU_WIDTH / 2), MENU_HEIGHT))
s = pygame.surface.Surface((WINDOW_WIDTH, (WINDOW_HEIGHT)/4))
s_dir = pygame.surface.Surface((WINDOW_WIDTH-10, WINDOW_HEIGHT-5))
s_dir_border = pygame.surface.Surface((WINDOW_WIDTH, (WINDOW_HEIGHT)))
s_proc = pygame.surface.Surface((WINDOW_WIDTH-10, WINDOW_HEIGHT-5))
s_proc_border = pygame.surface.Surface((WINDOW_WIDTH, (WINDOW_HEIGHT)))
s_info = pygame.surface.Surface(((WINDOW_WIDTH - 10), ((WINDOW_HEIGHT)/4) - 5))
s_graph_1 = pygame.surface.Surface((WINDOW_WIDTH, ((WINDOW_HEIGHT)/4)*2))

### Colors ###
red = (150,0,0)
```

```
green = (255,0,0)
blue = (0,0,255)
light_blue = (0,200,255)
black = (0,0,0)
white = (255, 255, 255)
grey = (150, 150, 150)
grey1 = (80, 80, 80)
grey2 = (45, 45, 45)
grey2 = (55, 55, 55)
grey_blue1 = (156, 181, 201)
grey_blue2 = (0, 55, 130)
dark_blue = (0, 0, 50)
sand = (255, 230, 185)
sand2 = (225, 190, 145)
dark_sand = (165, 145, 110)
background = (45, 45, 45)

### Fonts ###
pygame.font.init()
def make_font(fonts, size):
    available = pygame.font.get_fonts()
    # get_fonts() returns a list of lowercase spaceless font names
    choices = map(lambda x:x.lower().replace(' ', ''), fonts)
    for choice in choices:
        if choice in available:
            return pygame.font.SysFont(choice, size)
    return pygame.font.Font(None, size)
_cached_fonts = {}
def get_font(font_preferences, size):
    global _cached_fonts
    key = str(font_preferences) + '|' + str(size)
    font = _cached_fonts.get(key, None)
    if font == None:
        font = make_font(font_preferences, size)
        _cached_fonts[key] = font
    return font

font_preferences = ["Consolas","Rockwell", "Calibri", "Arial", ]

font_1 = get_font(font_preferences, 30)
font_2 = get_font(font_preferences, 20)
font_3 = get_font(font_preferences, 16)
font_4 = get_font(font_preferences, 12)

### Menu ###
### Menu Settings ###
pos_x = 125
pos_y = (int(MENU_BORDER_HEIGHT / 3) + 2)
button_size_pressed = 8
button_size_nonpressed = 10


def menu_buttons(menu):

    s_menu_border.fill(black)
    s_menu.fill(background)
    s_display.fill(background)

    #botton HOME
    pygame.draw.circle(s_display, black, (pos_x,pos_y), (button_size_nonpressed + 3))
    botton_display_0 = pygame.draw.circle(s_display, dark_sand, (pos_x,pos_y), button_size_nonpressed)
```

```
    #botton CPU
    pygame.draw.circle(s_display, black, ((pos_x + 50),pos_y), (button_size_nonpressed + 3))
    botton_display_1 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 50),pos_y),
button_size_nonpressed)
    #botton FILES
    pygame.draw.circle(s_display, black, ((pos_x + 100),pos_y), (button_size_nonpressed + 3))
    botton_display_2 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 100),pos_y),
button_size_nonpressed)
    #botton PROCESS
    pygame.draw.circle(s_display, black, ((pos_x + 150),pos_y), (button_size_nonpressed + 3))
    botton_display_3 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 150),pos_y),
button_size_nonpressed)

    if menu == 'HOME':
        pygame.draw.circle(s_display, black, (pos_x,pos_y), (button_size_pressed + 3))
        botton_display_0 = pygame.draw.circle(s_display, grey1, (pos_x,pos_y), button_size_pressed)
    elif menu == 'CPU':
        pygame.draw.circle(s_display, black, ((pos_x + 50),pos_y), (button_size_pressed + 3))
        botton_display_1 = pygame.draw.circle(s_display, grey1, ((pos_x + 50),pos_y), button_size_pressed)
    elif menu =='FILES':
        pygame.draw.circle(s_display, black, ((pos_x + 100),pos_y), (button_size_pressed + 3))
        botton_display_2 = pygame.draw.circle(s_display, grey1, ((pos_x + 100),pos_y), button_size_pressed)
    elif menu == 'PROCESS':
        pygame.draw.circle(s_display, black, ((pos_x + 150),pos_y), (button_size_pressed + 3))
        botton_display_3 = pygame.draw.circle(s_display, grey1, ((pos_x + 150),pos_y), button_size_pressed)

    pages = font_4.render("Tabs", 10, dark_sand)
    s_menu.blit(pages, (20, 10))
    ass = font_4.render("Dev.: Fábio R. P. Nunes", 10, dark_sand)
    s_menu.blit(ass, (600, 10))
    window.blit(s_menu_border, (0, WINDOW_HEIGHT))
    window.blit(s_menu, (5, (WINDOW_HEIGHT + 5)))
    window.blit(s_display, (200, (WINDOW_HEIGHT + 5)))

    return botton_display_0, botton_display_1, botton_display_2, botton_display_3
menu = 'HOME'
botton_display_0 = menu_buttons(menu)[0]
botton_display_1 = menu_buttons(menu)[1]
botton_display_2 = menu_buttons(menu)[2]
botton_display_3 = menu_buttons(menu)[3]
```

Arquivo: data_and_funcs.py

```
import psutil
import cpuinfo
import platform
import pygame
import socket
import os
import datetime
import display_interface

### Static Data ###
# PC Inf 1
pc_name = platform.node()
operational_system = platform.system()
osVersion = platform.version()
arc = platform.machine()
processor = platform.processor()
cores = psutil.cpu_count(logical=True)
pc_partitions = psutil.disk_partitions(all)
```

```
primary_storage = pc_partitions[0][0]
local = os.path.dirname(os.path.realpath(__file__))

# IPs
def get_ip_addresses(family):
    for interface, snics in psutil.net_if_addrs().items():
        for snic in snics:
            if snic.family == family:
                yield (interface, snic.address)

ipv4s = list(get_ip_addresses(socket.AF_INET))
ipv6s = list(get_ip_addresses(socket.AF_INET6))
mac_address = list(get_ip_addresses(psutil.AF_LINK))

### Dynamic Data ###
def cpu_graph_0():
    ### DATA ###
    cpu = int(psutil.cpu_percent(interval=0.1))
    freq_per_cent = int(psutil.cpu_percent(interval=0.1))
    freq_max = psutil.cpu_freq().max
    freq_current = ((freq_max * freq_per_cent)/100)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*cpu) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,54), (790, 54), 6)

    bar_text = "CPU Usage"
    bar_text_2 = "- Current Frequency (Mhz): " + str(freq_current)
    bar_text_inside_01 = str(cpu) + "%"
    bar_text_inside_02 = str((100 - cpu)) + "%"


    text = display_interface.font_1.render(bar_text, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)

    display_interface.s.blit(text, (20, 18))
    display_interface.s.blit(text_2, (185, 26))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (700, 110))

    display_interface.window.blit(display_interface.s, (0, 0))

def cpu_graph_1():
    ### Data ###
    cores_percent = psutil.cpu_percent(interval=1, percpu=True)

    display_interface.s_graph_1.fill(display_interface.background)

    cores = len(cores_percent)
```

```
    x = y = shift = 10

    height = int(display_interface.s_graph_1.get_height() - 2*y)
    width = int((display_interface.s_graph_1.get_width() - 2*y - (cores + 1)*shift ) / cores)

    d = x + shift
    cores_count = 1
    for i in cores_percent:
        pygame.draw.rect(display_interface.s_graph_1, display_interface.black, (d+5, y+5, width, height))
        pygame.draw.rect(display_interface.s_graph_1, display_interface.red, (d, y, width, height))
        pygame.draw.rect(display_interface.s_graph_1, display_interface.grey_blue2, (d, y, width, ((1-
i/100)*height)))
        bar_text_inside_0 = str(int(i)) + "%"
        text_inside_0 = display_interface.font_3.render(bar_text_inside_0, 10, display_interface.sand2)
        display_interface.s_graph_1.blit(text_inside_0, ((d + (width/3)), (height - 10)))
        bar_text_inside_1 = "#"+str(cores_count)
        text_inside_1 = display_interface.font_3.render(bar_text_inside_1, 10, display_interface.dark_sand)
        display_interface.s_graph_1.blit(text_inside_1, ((d + (width/3)), y+10))
        cores_count += 1
        d = d + width + shift

    display_interface.window.blit(display_interface.s_graph_1, (0, (display_interface.WINDOW_HEIGHT)/4))

def mem_graph_0():
    ### DATA ###
    mem = psutil.virtual_memory()
    memTotalGigas = round((mem.total / (1024*1024*1024)), 2)
    memUsedGigas = round((mem.used / (1024*1024*1024)), 2)
    memFreeGigas = round((mem.free / (1024*1024*1024)), 2)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*mem.percent) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH - 10), 45), 6)

    bar_text_1 = "Memory Usage"
    bar_text_2 = "(Total: " + str(memTotalGigas) + " GB)"
    bar_text_inside_01 = str(memUsedGigas) + " Used"
    bar_text_inside_02 = str(memFreeGigas) + " Free"

    text_1 = display_interface.font_1.render(bar_text_1, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)

    display_interface.s.blit(text_1, (20, 10))
    display_interface.s.blit(text_2, (235, 18))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (655, 110))

    display_interface.window.blit(display_interface.s, (0, ((display_interface.WINDOW_HEIGHT)/4)))
```

```
def hd_graph_0():
    ### DATA ###
    hd = psutil.disk_usage(str(primary_storage))
    hdTotalGigas = round((hd.total / (1024*1024*1024)), 2)
    hdUsedGigas = round((hd.used / (1024*1024*1024)), 2)
    hdFreeGigas = round((hd.free / (1024*1024*1024)), 2)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*hd.percent) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,45), (790, 45), 6)

    bar_text_1 = "Primary Storage Usage"
    bar_text_2 = "(Total: " + str(hdTotalGigas) + " GB)"
    bar_text_inside_01 = str(hdUsedGigas) + " Used"
    bar_text_inside_02 = str(hdFreeGigas) + " Free"

    text_1 = display_interface.font_1.render(bar_text_1, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)

    display_interface.s.blit(text_1, (20, 10))
    display_interface.s.blit(text_2, (390, 18))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (640, 110))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*2))))

def pc_1():
    display_interface.s_info.fill(display_interface.background)
    display_interface.s.fill(display_interface.black)

    pygame.draw.line(display_interface.s_info, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH -20), 45), 6)
    pos_y = 10
    text_0 = "PC Information"
    text_1 = "PC Name: " + str(pc_name)
    text_2 = "Operating System: " + str(operational_system) + " " + str(osVersion)
    text_3 = "Networking Information"

    ipv4s_text_pos_y = 85
    text_4 = "- IPv4s"
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)
    display_interface.s_info.blit(font_text_4, (20, ipv4s_text_pos_y))
    ipv4s_text_pos_y += 15
    for i in range(len(ipv4s)):
        text = ipv4s[i][0] + ": " + ipv4s[i][1]
        font_text = display_interface.font_4.render(text, 10, display_interface.dark_sand)
        display_interface.s_info.blit(font_text, (20, ipv4s_text_pos_y))
        ipv4s_text_pos_y += 15

    text_4 = "MAC Address: " + str(mac_address[0][1])
```

```
    ipv6s_text_pos_y = 55
    text_5 = "- IPv6s"
    font_text_5 = display_interface.font_4.render(text_5, 10, display_interface.dark_sand)
    display_interface.s_info.blit(font_text_5, (400, ipv6s_text_pos_y))
    ipv6s_text_pos_y += 15
    for i in range(len(ipv6s)):
        text = ipv6s[i][0] + ": " + ipv6s[i][1]
        font_text = display_interface.font_4.render(text, 10, display_interface.dark_sand)
        display_interface.s_info.blit(font_text, (400, ipv6s_text_pos_y))
        ipv6s_text_pos_y += 15

    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    font_text_1 = display_interface.font_4.render(text_1, 10, display_interface.grey)
    font_text_2 = display_interface.font_4.render(text_2, 10, display_interface.grey)
    font_text_3 = display_interface.font_4.render(text_3, 10, display_interface.dark_sand)
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)

    display_interface.s_info.blit(font_text_0, (20, pos_y))
    display_interface.s_info.blit(font_text_1, (275, pos_y))
    pos_y += 15
    display_interface.s_info.blit(font_text_2, (275, pos_y))
    pos_y += 30
    display_interface.s_info.blit(font_text_3, (20, pos_y))
    display_interface.s_info.blit(font_text_4, (20, 70))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*3))))
    display_interface.window.blit(display_interface.s_info, (5,
(((((display_interface.WINDOW_HEIGHT)/4)*3)+5))))

def cpu_inf():
    inf = cpuinfo.get_cpu_info()
    cpu_name = inf['brand_raw']
    cpu_arch = inf['arch']
    cpu_bits = inf['bits']

    p_cores = psutil.cpu_count(logical=False)
    l_cores = psutil.cpu_count()

    display_interface.s_info.fill(display_interface.background)
    display_interface.s.fill(display_interface.black)

    pygame.draw.line(display_interface.s_info, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH -20), 45), 6)

    text_0 = "CPU Information"
    text_1 = "CPU Name: " + str(cpu_name)
    text_2 = "Architecture: " + str(cpu_arch) + " (" + str(arc) + ")"
    text_3 = "Word Length: " + str(cpu_bits) + " bits"
    text_4 = "Physical Cores: " + str(p_cores)
    text_5 = "Logical Cores: " + str(l_cores)

    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    font_text_1 = display_interface.font_4.render(text_1, 10, display_interface.dark_sand)
    font_text_2 = display_interface.font_4.render(text_2, 10, display_interface.dark_sand)
    font_text_3 = display_interface.font_4.render(text_3, 10, display_interface.dark_sand)
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)
    font_text_5 = display_interface.font_4.render(text_5, 10, display_interface.dark_sand)

    display_interface.s_info.blit(font_text_0, (20, 10))
    display_interface.s_info.blit(font_text_1, (20, 55))
```

```
    display_interface.s_info.blit(font_text_2, (20, 70))
    display_interface.s_info.blit(font_text_3, (20, 85))
    display_interface.s_info.blit(font_text_4, (20, 100))
    display_interface.s_info.blit(font_text_5, (20, 115))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*3))))
    display_interface.window.blit(display_interface.s_info, (5,
(((((display_interface.WINDOW_HEIGHT)/4)*3)+5))))

def dir_files():
    lista = os.listdir(local)
    files = {}
    for i in lista:
        filepath = os.path.join(local, i)
        if os.path.isfile(filepath):
            files[i] = {}
            files[i]['Size'] = os.stat(filepath).st_size
            files[i]['Access'] = datetime.datetime.fromtimestamp(os.stat(filepath).st_atime)
            files[i]['Mod'] = datetime.datetime.fromtimestamp(os.stat(filepath).st_mtime)
    return files

def files_page(list_files):
    files = list_files
    display_interface.s_dir.fill(display_interface.background)
    display_interface.s_dir_border.fill(display_interface.black)
    pos_x = 20
    pos_y = 10
    text_0 = "Local Files"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_dir.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = str(local)
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_dir.blit(font_text_1, (pos_x, pos_y))
    pos_y += 28

    pygame.draw.line(display_interface.s_dir, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)

    pos_x_files = 47
    pos_y_files = 78
    files_count = 0
    for i in list(files):
        if files_count < 12:
            file_text = '{:^0.34}'.format(str(i))
            file_size_text = "Size: " + '{:^0.38}'.format(str(files[i]['Size']) + " bytes")
            file_created_text = "Access: " + '{:^0.38}'.format(str(files[i]['Access']))
            file_mod_text = "Modification: "+ '{:^0.38}'.format(str(files[i]['Mod']))

            font_file_text = display_interface.font_3.render(file_text, 10, display_interface.dark_sand)
            font_file_size_text = display_interface.font_4.render(file_size_text, 10,
display_interface.dark_sand)
            font_file_created_text = display_interface.font_4.render(file_created_text, 10,
display_interface.dark_sand)
            font_file_mod_text = display_interface.font_4.render(file_mod_text, 10,
display_interface.dark_sand)

            pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files, pos_y_files+3,
int((display_interface.WINDOW_WIDTH)/2)-75, (73)))
            pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files-5, pos_y_files-
4, int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
```

```
            pygame.draw.rect(display_interface.s_dir, display_interface.grey2, (pos_x_files-4, pos_y_files-
3, int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
            display_interface.s_dir.blit(font_file_text, (pos_x_files + 8, pos_y_files))
            pygame.draw.circle(display_interface.s_dir, display_interface.black, ((pos_x_files-
5),(pos_y_files + 8)), 8)
            botton_file = pygame.draw.circle(display_interface.s_dir, display_interface.grey_blue2,
((pos_x_files-5),(pos_y_files + 8)), 6)
            pos_y_files += 18
            pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files-5, pos_y_files,
int((display_interface.WINDOW_WIDTH)/2)-75, (53)))
            pygame.draw.rect(display_interface.s_dir, display_interface.grey2, (pos_x_files-4,
pos_y_files+1, int((display_interface.WINDOW_WIDTH)/2)-77, (51)))
            pos_y_files += 5
            display_interface.s_dir.blit(font_file_size_text, (pos_x_files, pos_y_files))
            pos_y_files += 15
            display_interface.s_dir.blit(font_file_created_text, (pos_x_files, pos_y_files))
            pos_y_files += 15
            display_interface.s_dir.blit(font_file_mod_text, (pos_x_files, pos_y_files))
            pos_y_files += 30
            files.pop(i)
            files_count += 1
            if (files_count == 6):
                pos_x_files += int(((display_interface.WINDOW_WIDTH)/2)-22)
                pos_y_files = 78
    display_interface.window.blit(display_interface.s_dir_border, (0, 0))
    display_interface.window.blit(display_interface.s_dir, (5, 5))
    return files

def files_next_page(list_left_files):
    files = list_left_files
    if len(files) > 0:
        text_next_page = "Next Page"
        font_text_next_page = display_interface.font_3.render(text_next_page, 10,
display_interface.dark_sand)
        display_interface.s_dir.blit(font_text_next_page, ((display_interface.WINDOW_WIDTH-110), 10))
        pygame.draw.circle(display_interface.s_dir, display_interface.black,
((display_interface.WINDOW_WIDTH-70),40), (10 + 3))
        botton_next_files = pygame.draw.circle(display_interface.s_dir, display_interface.dark_sand,
((display_interface.WINDOW_WIDTH-70),40), 10)
        display_interface.window.blit(display_interface.s_dir_border, (0, 0))
        display_interface.window.blit(display_interface.s_dir, (5, 5))
        return True
    elif len(files) <= 0:
        return False

def running_processes():
    processes = psutil.pids()
    process = {}
    for i in processes:
        try:
            proc_name = str(psutil.Process(i).name())
            if proc_name not in process:
                process[proc_name] = {}
                process[proc_name]['PIDs'] = []
                process[proc_name]['Status'] = []
                process[proc_name]['# Total'] = 0
            process[proc_name]['PIDs'].append(i)
            process[proc_name]['Status'].append(psutil.Process(i).status())
            process[proc_name]['# Total'] += 1
        except (psutil.AccessDenied, psutil.ZombieProcess):
            pass
```

```
        except psutil.NoSuchProcess:
            continue
    return process

def processes_page(list_process):
    process = list_process
    display_interface.s_proc.fill(display_interface.background)
    display_interface.s_proc_border.fill(display_interface.black)
    pos_x = 20
    pos_y = 10
    text_0 = "Process"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_proc.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = "Total of: " + str(len(process)) + " process"
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_proc.blit(font_text_1, (pos_x, pos_y))

    pygame.draw.line(display_interface.s_proc, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)

    pos_x_process = 47
    pos_y_process = 78
    process_count = 0
    for i in list(process):
        if process_count < 12:
            process_text = str(i)
            process_size_text = "PIDs: " + '{:^0.38}'.format(str(process[i]['PIDs']))
            process_created_text = "Status: " + '{:^0.37}'.format(str(process[i]['Status']))
            process_mod_text = "# Total: "+ '{:^0.38}'.format(str(process[i]['# Total']))

            font_process_text = display_interface.font_3.render(process_text, 10,
display_interface.dark_sand)
            font_process_size_text = display_interface.font_4.render(process_size_text, 10,
display_interface.dark_sand)
            font_process_created_text = display_interface.font_4.render(process_created_text, 10,
display_interface.dark_sand)
            font_process_mod_text = display_interface.font_4.render(process_mod_text, 10,
display_interface.dark_sand)

            pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process,
pos_y_process+3, int((display_interface.WINDOW_WIDTH)/2)-75, (73)))
            pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process-5,
pos_y_process-4, int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
            pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_process-4,
pos_y_process-3, int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
            display_interface.s_proc.blit(font_process_text, (pos_x_process + 8, pos_y_process))
            pygame.draw.circle(display_interface.s_proc, display_interface.black, ((pos_x_process-
5),(pos_y_process + 8)), 8)
            botton_process = pygame.draw.circle(display_interface.s_proc, display_interface.grey_blue2,
((pos_x_process-5),(pos_y_process + 8)), 6)
            pos_y_process += 18
            pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process-5,
pos_y_process, int((display_interface.WINDOW_WIDTH)/2)-75, (53)))
            pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_process-4,
pos_y_process+1, int((display_interface.WINDOW_WIDTH)/2)-77, (51)))
            pos_y_process += 5
            display_interface.s_proc.blit(font_process_size_text, (pos_x_process, pos_y_process))
            pos_y_process += 15
            display_interface.s_proc.blit(font_process_created_text, (pos_x_process, pos_y_process))
            pos_y_process += 15
```

```
            display_interface.s_proc.blit(font_process_mod_text, (pos_x_process, pos_y_process))
            pos_y_process += 30
            process.pop(i)

            process_count += 1
            if (process_count == 6):
                pos_x_process += int(((display_interface.WINDOW_WIDTH)/2)-22)
                pos_y_process = 78
        display_interface.window.blit(display_interface.s_proc_border, (0, 0))
        display_interface.window.blit(display_interface.s_proc, (5, 5))
    return process

def process_next_page(list_left_process):
    process = list_left_process
    if len(process) > 0:
        text_next_page = "Next Page"
        font_text_next_page = display_interface.font_3.render(text_next_page, 10,
display_interface.dark_sand)
        display_interface.s_proc.blit(font_text_next_page, ((display_interface.WINDOW_WIDTH-110), 10))
        pygame.draw.circle(display_interface.s_proc, display_interface.black,
((display_interface.WINDOW_WIDTH-70),40), (10 + 3))
        botton_next_process = pygame.draw.circle(display_interface.s_proc, display_interface.dark_sand,
((display_interface.WINDOW_WIDTH-70),40), 10)
        display_interface.window.blit(display_interface.s_proc_border, (0, 0))
        display_interface.window.blit(display_interface.s_proc, (5, 5))
        return True
    elif len(process) <= 0:
        return False
```

Código da Versão 4.0

Arquivo: main.py

```
import pygame
import data_and_funcs
import display_interface
import time
import sched
import threading

scheduler = sched.scheduler(time.time, time.sleep)

def main():
    pygame.display.set_caption("PC Resources Monitor")
    pygame.display.init()

    ### Display ###
    clock = pygame.time.Clock()
    cont = 0
    close = False
    files_first_page = True
    process_first_page = True
    counter_change_page = 0
    while not close:
        t1 = time.perf_counter()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                close = True
            ### Keyboard ###
            if event.type == pygame.KEYDOWN:
                if ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'HOME')) or ((event.key ==
pygame.K_LEFT) and (display_interface.menu == 'FILES')):
                    display_interface.menu = 'CPU'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'CPU')) or ((event.key
== pygame.K_LEFT) and (display_interface.menu == 'PROCESS')):
                    display_interface.menu = 'FILES'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'FILES')) or ((event.key
== pygame.K_LEFT) and (display_interface.menu == 'HOME')):
                    display_interface.menu = 'PROCESS'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'PROCESS')) or
((event.key == pygame.K_LEFT) and (display_interface.menu == 'CPU')) or ((event.key == pygame.K_SPACE)):
                    display_interface.menu = 'HOME'
            ### Mouse ###
            if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                pos = pygame.mouse.get_pos()
                if display_interface.botton_display_0.collidepoint(((pos[0] - 200), (pos[1] -
(display_interface.WINDOW_HEIGHT + 5)))):
                    display_interface.menu = 'HOME'
                elif display_interface.botton_display_1.collidepoint(((pos[0] - 200), (pos[1] -
(display_interface.WINDOW_HEIGHT + 5)))):
                    display_interface.menu = 'CPU'
                elif display_interface.botton_display_2.collidepoint(((pos[0] - 200), (pos[1] -
(display_interface.WINDOW_HEIGHT + 5)))):
                    display_interface.menu = 'FILES'
                elif display_interface.botton_display_3.collidepoint(((pos[0] - 200), (pos[1] -
(display_interface.WINDOW_HEIGHT + 5)))):
                    display_interface.menu = 'PROCESS'
        cont += 1
        if cont == 1:
            display_interface.menu_buttons(display_interface.menu)
            if display_interface.menu == 'HOME':
```

```
                #t1 = time.perf_counter()
                data_and_funcs.cpu_graph_0()
                data_and_funcs.mem_graph_0()
                data_and_funcs.hd_graph_0()
                data_and_funcs.pc_1()
                t2 = time.perf_counter()
                print("Process Time on Home Page: " + str(t2 - t1))
                counter_change_page += 1
                #if counter_change_page == 5:
                    #display_interface.menu = 'CPU'
                    #counter_change_page = 0

            elif display_interface.menu == 'CPU':
                #t1 = time.perf_counter()
                thrad_1 = threading.Thread(target=data_and_funcs.cpu_graph_0)
                thrad_2 = threading.Thread(target=data_and_funcs.cpu_graph_1)
                thrad_3 = threading.Thread(target=data_and_funcs.cpu_inf)
                thrad_1.start()
                thrad_2.start()
                thrad_3.start()
                thrad_1.join()
                thrad_2.join()
                thrad_3.join()
                t2 = time.perf_counter()
                print("Process Time on CPU Page: " + str(t2 - t1))
                counter_change_page += 1
                #if counter_change_page == 5:
                    #display_interface.menu = 'FILES'
                    #counter_change_page = 0

            elif display_interface.menu =='FILES':
                #t1 = time.perf_counter()
                files = data_and_funcs.dir_files()
                if files_first_page:
                    files_left = data_and_funcs.files_page(files)
                if data_and_funcs.files_next_page(files_left):
                    if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                        pos = pygame.mouse.get_pos()
                        if (pos[0] - 205) >= (520) and (pos[0] - 205) <= (540) and (pos[1] - 5) >= (30) and
(pos[1] - 5) <= (50):
                            files_left = data_and_funcs.files_page(files_left)
                            files_first_page = False
                t2 = time.perf_counter()
                print("Process Time on FILES Page: " + str(t2 - t1))
                counter_change_page += 1
                #if counter_change_page == 5:
                    #display_interface.menu = 'PROCESS'
                    #counter_change_page = 0
            elif display_interface.menu == 'PROCESS':
                #t1 = time.perf_counter()
                process = data_and_funcs.running_processes()
                if process_first_page:
                    process_left = data_and_funcs.processes_page(process)
                if data_and_funcs.process_next_page(process_left):
                    if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                        pos = pygame.mouse.get_pos()
                        if (pos[0] - 205) >= (520) and (pos[0] - 205) <= (540) and (pos[1] - 5) >= (30) and
(pos[1] - 5) <= (50):
                            process_left = data_and_funcs.processes_page(process_left)
                            process_first_page = False
                t2 = time.perf_counter()
```

```
                print("Process Time on PROCESS Page: " + str(t2 - t1))
                counter_change_page += 1
                #if counter_change_page == 5:
                    #display_interface.menu = 'HOME'
                    #counter_change_page = 0
            cont = 0
        pygame.display.update()
        clock.tick(60)
    pygame.display.quit()
if __name__ == "__main__":
    main()
```

Arquivo: display_interface.py

```
import pygame
import main

### Windows Config. ###
WINDOW_WIDTH = 800
WINDOW_HEIGHT = 600
MENU_BORDER_WIDTH = WINDOW_WIDTH
MENU_BORDER_HEIGHT = 40
MENU_WIDTH = (WINDOW_WIDTH - 10)
MENU_HEIGHT = (MENU_BORDER_HEIGHT - 10)

window = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT + MENU_BORDER_HEIGHT))

### Surfaces ###
s_menu = pygame.surface.Surface((MENU_WIDTH, MENU_HEIGHT))
s_menu_border = pygame.surface.Surface((MENU_BORDER_WIDTH, MENU_BORDER_HEIGHT))
s_display = pygame.surface.Surface(((MENU_WIDTH / 2), MENU_HEIGHT))
s = pygame.surface.Surface((WINDOW_WIDTH, (WINDOW_HEIGHT)/4))
s_dir = pygame.surface.Surface((WINDOW_WIDTH-10, WINDOW_HEIGHT-5))
s_dir_border = pygame.surface.Surface((WINDOW_WIDTH, (WINDOW_HEIGHT)))
s_proc = pygame.surface.Surface((WINDOW_WIDTH-10, WINDOW_HEIGHT-5))
s_proc_border = pygame.surface.Surface((WINDOW_WIDTH, (WINDOW_HEIGHT)))
s_info = pygame.surface.Surface(((WINDOW_WIDTH - 10), ((WINDOW_HEIGHT)/4) - 5))
s_graph_1 = pygame.surface.Surface((WINDOW_WIDTH, ((WINDOW_HEIGHT)/4)*2))

### Colors ###
red = (150,0,0)
green = (255,0,0)
blue = (0,0,255)
light_blue = (0,200,255)
black = (0,0,0)
white = (255, 255, 255)
grey = (150, 150, 150)
grey1 = (80, 80, 80)
grey2 = (45, 45, 45)
grey2 = (55, 55, 55)
grey_blue1 = (156, 181, 201)
grey_blue2 = (0, 55, 130)
dark_blue = (0, 0, 50)
sand = (255, 230, 185)
sand2 = (225, 190, 145)
dark_sand = (165, 145, 110)
background = (45, 45, 45)

### Fonts ###
pygame.font.init()
def make_font(fonts, size):
```

```
    available = pygame.font.get_fonts()
    # get_fonts() returns a list of lowercase spaceless font names
    choices = map(lambda x:x.lower().replace(' ', ''), fonts)
    for choice in choices:
        if choice in available:
            return pygame.font.SysFont(choice, size)
    return pygame.font.Font(None, size)
_cached_fonts = {}
def get_font(font_preferences, size):
    global _cached_fonts
    key = str(font_preferences) + '|' + str(size)
    font = _cached_fonts.get(key, None)
    if font == None:
        font = make_font(font_preferences, size)
        _cached_fonts[key] = font
    return font

font_preferences = ["Consolas","Rockwell", "Calibri", "Arial", ]

font_1 = get_font(font_preferences, 30)
font_2 = get_font(font_preferences, 20)
font_3 = get_font(font_preferences, 16)
font_4 = get_font(font_preferences, 12)

### Menu ###
### Menu Settings ###
pos_x = 125
pos_y = (int(MENU_BORDER_HEIGHT / 3) + 2)
button_size_pressed = 8
button_size_nonpressed = 10


def menu_buttons(menu):

    s_menu_border.fill(black)
    s_menu.fill(background)
    s_display.fill(background)

    #botton HOME
    pygame.draw.circle(s_display, black, (pos_x,pos_y), (button_size_nonpressed + 3))
    botton_display_0 = pygame.draw.circle(s_display, dark_sand, (pos_x,pos_y), button_size_nonpressed)
    #botton CPU
    pygame.draw.circle(s_display, black, ((pos_x + 50),pos_y), (button_size_nonpressed + 3))
    botton_display_1 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 50),pos_y),
button_size_nonpressed)
    #botton FILES
    pygame.draw.circle(s_display, black, ((pos_x + 100),pos_y), (button_size_nonpressed + 3))
    botton_display_2 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 100),pos_y),
button_size_nonpressed)
    #botton PROCESS
    pygame.draw.circle(s_display, black, ((pos_x + 150),pos_y), (button_size_nonpressed + 3))
    botton_display_3 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 150),pos_y),
button_size_nonpressed)

    if menu == 'HOME':
        pygame.draw.circle(s_display, black, (pos_x,pos_y), (button_size_pressed + 3))
        botton_display_0 = pygame.draw.circle(s_display, grey1, (pos_x,pos_y), button_size_pressed)
    elif menu == 'CPU':
        pygame.draw.circle(s_display, black, ((pos_x + 50),pos_y), (button_size_pressed + 3))
        botton_display_1 = pygame.draw.circle(s_display, grey1, ((pos_x + 50),pos_y), button_size_pressed)
    elif menu =='FILES':
```

```
        pygame.draw.circle(s_display, black, ((pos_x + 100),pos_y), (button_size_pressed + 3))
        botton_display_2 = pygame.draw.circle(s_display, grey1, ((pos_x + 100),pos_y), button_size_pressed)
    elif menu == 'PROCESS':
        pygame.draw.circle(s_display, black, ((pos_x + 150),pos_y), (button_size_pressed + 3))
        botton_display_3 = pygame.draw.circle(s_display, grey1, ((pos_x + 150),pos_y), button_size_pressed)

    pages = font_4.render("Tabs", 10, dark_sand)
    s_menu.blit(pages, (20, 10))
    ass = font_4.render("Dev.: Fábio R. P. Nunes", 10, dark_sand)
    s_menu.blit(ass, (600, 10))
    window.blit(s_menu_border, (0, WINDOW_HEIGHT))
    window.blit(s_menu, (5, (WINDOW_HEIGHT + 5)))
    window.blit(s_display, (200, (WINDOW_HEIGHT + 5)))

    return botton_display_0, botton_display_1, botton_display_2, botton_display_3
menu = 'HOME'
botton_display_0 = menu_buttons(menu)[0]
botton_display_1 = menu_buttons(menu)[1]
botton_display_2 = menu_buttons(menu)[2]
botton_display_3 = menu_buttons(menu)[3]
```

Arquivo: data_and_funcs.py

```
import psutil
import cpuinfo
import platform
import pygame
import socket
import os
import datetime
import display_interface

### Static Data ###
# PC Inf 1
pc_name = platform.node()
operational_system = platform.system()
osVersion = platform.version()
arc = platform.machine()
processor = platform.processor()
cores = psutil.cpu_count(logical=True)
pc_partitions = psutil.disk_partitions(all)
primary_storage = pc_partitions[0][0]
local = os.path.dirname(os.path.realpath(__file__))

# IPs
def get_ip_addresses(family):
    for interface, snics in psutil.net_if_addrs().items():
        for snic in snics:
            if snic.family == family:
                yield (interface, snic.address)

ipv4s = list(get_ip_addresses(socket.AF_INET))
ipv6s = list(get_ip_addresses(socket.AF_INET6))
mac_address = list(get_ip_addresses(psutil.AF_LINK))

### Dynamic Data ###
def cpu_graph_0():
    ### DATA ###
    cpu = int(psutil.cpu_percent(interval=0.1))
    freq_per_cent = int(psutil.cpu_percent(interval=0.1))
    freq_max = psutil.cpu_freq().max
```

```
    freq_current = ((freq_max * freq_per_cent)/100)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*cpu) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,54), (790, 54), 6)

    bar_text = "CPU Usage"
    bar_text_2 = "- Current Frequency (Mhz): " + str(freq_current)
    bar_text_inside_01 = str(cpu) + "%"
    bar_text_inside_02 = str((100 - cpu)) + "%"


    text = display_interface.font_1.render(bar_text, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)

    display_interface.s.blit(text, (20, 18))
    display_interface.s.blit(text_2, (185, 26))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (700, 110))

    display_interface.window.blit(display_interface.s, (0, 0))

def cpu_graph_1():
    ### Data ###
    cores_percent = psutil.cpu_percent(interval=1, percpu=True)

    display_interface.s_graph_1.fill(display_interface.background)

    cores = len(cores_percent)

    x = y = shift = 10

    height = int(display_interface.s_graph_1.get_height() - 2*y)
    width = int((display_interface.s_graph_1.get_width() - 2*y - (cores + 1)*shift ) / cores)

    d = x + shift
    cores_count = 1
    for i in cores_percent:
        pygame.draw.rect(display_interface.s_graph_1, display_interface.black, (d+5, y+5, width, height))
        pygame.draw.rect(display_interface.s_graph_1, display_interface.red, (d, y, width, height))
        pygame.draw.rect(display_interface.s_graph_1, display_interface.grey_blue2, (d, y, width, ((1-
i/100)*height)))
        bar_text_inside_0 = str(int(i)) + "%"
        text_inside_0 = display_interface.font_3.render(bar_text_inside_0, 10, display_interface.sand2)
        display_interface.s_graph_1.blit(text_inside_0, ((d + (width/3)), (height - 10)))
        bar_text_inside_1 = "#"+str(cores_count)
        text_inside_1 = display_interface.font_3.render(bar_text_inside_1, 10, display_interface.dark_sand)
        display_interface.s_graph_1.blit(text_inside_1, ((d + (width/3)), y+10))
        cores_count += 1
        d = d + width + shift
```

```
    display_interface.window.blit(display_interface.s_graph_1, (0, (display_interface.WINDOW_HEIGHT)/4))

def mem_graph_0():
    ### DATA ###
    mem = psutil.virtual_memory()
    memTotalGigas = round((mem.total / (1024*1024*1024)), 2)
    memUsedGigas = round((mem.used / (1024*1024*1024)), 2)
    memFreeGigas = round((mem.free / (1024*1024*1024)), 2)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*mem.percent) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH - 10), 45), 6)

    bar_text_1 = "Memory Usage"
    bar_text_2 = "(Total: " + str(memTotalGigas) + " GB)"
    bar_text_inside_01 = str(memUsedGigas) + " Used"
    bar_text_inside_02 = str(memFreeGigas) + " Free"

    text_1 = display_interface.font_1.render(bar_text_1, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)

    display_interface.s.blit(text_1, (20, 10))
    display_interface.s.blit(text_2, (235, 18))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (655, 110))

    display_interface.window.blit(display_interface.s, (0, ((display_interface.WINDOW_HEIGHT)/4)))

def hd_graph_0():
    ### DATA ###
    hd = psutil.disk_usage(str(primary_storage))
    hdTotalGigas = round((hd.total / (1024*1024*1024)), 2)
    hdUsedGigas = round((hd.used / (1024*1024*1024)), 2)
    hdFreeGigas = round((hd.free / (1024*1024*1024)), 2)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*hd.percent) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,45), (790, 45), 6)
```

```
    bar_text_1 = "Primary Storage Usage"
    bar_text_2 = "(Total: " + str(hdTotalGigas) + " GB)"
    bar_text_inside_01 = str(hdUsedGigas) + " Used"
    bar_text_inside_02 = str(hdFreeGigas) + " Free"

    text_1 = display_interface.font_1.render(bar_text_1, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)

    display_interface.s.blit(text_1, (20, 10))
    display_interface.s.blit(text_2, (390, 18))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (640, 110))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*2))))

def pc_1():
    display_interface.s_info.fill(display_interface.background)
    display_interface.s.fill(display_interface.black)

    pygame.draw.line(display_interface.s_info, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH -20), 45), 6)
    pos_y = 10
    text_0 = "PC Information"
    text_1 = "PC Name: " + str(pc_name)
    text_2 = "Operating System: " + str(operational_system) + " " + str(osVersion)
    text_3 = "Networking Information"

    ipv4s_text_pos_y = 85
    text_4 = "- IPv4s"
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)
    display_interface.s_info.blit(font_text_4, (20, ipv4s_text_pos_y))
    ipv4s_text_pos_y += 15
    for i in range(len(ipv4s)):
        text = ipv4s[i][0] + ": " + ipv4s[i][1]
        font_text = display_interface.font_4.render(text, 10, display_interface.dark_sand)
        display_interface.s_info.blit(font_text, (20, ipv4s_text_pos_y))
        ipv4s_text_pos_y += 15

    text_4 = "MAC Address: " + str(mac_address[0][1])

    ipv6s_text_pos_y = 55
    text_5 = "- IPv6s"
    font_text_5 = display_interface.font_4.render(text_5, 10, display_interface.dark_sand)
    display_interface.s_info.blit(font_text_5, (400, ipv6s_text_pos_y))
    ipv6s_text_pos_y += 15
    for i in range(len(ipv6s)):
        text = ipv6s[i][0] + ": " + ipv6s[i][1]
        font_text = display_interface.font_4.render(text, 10, display_interface.dark_sand)
        display_interface.s_info.blit(font_text, (400, ipv6s_text_pos_y))
        ipv6s_text_pos_y += 15

    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    font_text_1 = display_interface.font_4.render(text_1, 10, display_interface.grey)
    font_text_2 = display_interface.font_4.render(text_2, 10, display_interface.grey)
    font_text_3 = display_interface.font_4.render(text_3, 10, display_interface.dark_sand)
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)

    display_interface.s_info.blit(font_text_0, (20, pos_y))
    display_interface.s_info.blit(font_text_1, (275, pos_y))
```

```
    pos_y += 15
    display_interface.s_info.blit(font_text_2, (275, pos_y))
    pos_y += 30
    display_interface.s_info.blit(font_text_3, (20, pos_y))
    display_interface.s_info.blit(font_text_4, (20, 70))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*3))))
    display_interface.window.blit(display_interface.s_info, (5,
(((((display_interface.WINDOW_HEIGHT)/4)*3)+5))))

def cpu_inf():
    inf = cpuinfo.get_cpu_info()
    cpu_name = inf['brand_raw']
    cpu_arch = inf['arch']
    cpu_bits = inf['bits']

    p_cores = psutil.cpu_count(logical=False)
    l_cores = psutil.cpu_count()

    display_interface.s_info.fill(display_interface.background)
    display_interface.s.fill(display_interface.black)

    pygame.draw.line(display_interface.s_info, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH -20), 45), 6)

    text_0 = "CPU Information"
    text_1 = "CPU Name: " + str(cpu_name)
    text_2 = "Architecture: " + str(cpu_arch) + " (" + str(arc) + ")"
    text_3 = "Word Length: " + str(cpu_bits) + " bits"
    text_4 = "Physical Cores: " + str(p_cores)
    text_5 = "Logical Cores: " + str(l_cores)

    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    font_text_1 = display_interface.font_4.render(text_1, 10, display_interface.dark_sand)
    font_text_2 = display_interface.font_4.render(text_2, 10, display_interface.dark_sand)
    font_text_3 = display_interface.font_4.render(text_3, 10, display_interface.dark_sand)
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)
    font_text_5 = display_interface.font_4.render(text_5, 10, display_interface.dark_sand)

    display_interface.s_info.blit(font_text_0, (20, 10))
    display_interface.s_info.blit(font_text_1, (20, 55))
    display_interface.s_info.blit(font_text_2, (20, 70))
    display_interface.s_info.blit(font_text_3, (20, 85))
    display_interface.s_info.blit(font_text_4, (20, 100))
    display_interface.s_info.blit(font_text_5, (20, 115))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*3))))
    display_interface.window.blit(display_interface.s_info, (5,
(((((display_interface.WINDOW_HEIGHT)/4)*3)+5))))

def dir_files():
    lista = os.listdir(local)
    files = {}
    for i in lista:
        filepath = os.path.join(local, i)
        if os.path.isfile(filepath):
            files[i] = {}
            files[i]['Size'] = os.stat(filepath).st_size
            files[i]['Access'] = datetime.datetime.fromtimestamp(os.stat(filepath).st_atime)
            files[i]['Mod'] = datetime.datetime.fromtimestamp(os.stat(filepath).st_mtime)
    return files
```

```
def files_page(list_files):
    files = list_files
    display_interface.s_dir.fill(display_interface.background)
    display_interface.s_dir_border.fill(display_interface.black)
    pos_x = 20
    pos_y = 10
    text_0 = "Local Files"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_dir.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = str(local)
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_dir.blit(font_text_1, (pos_x, pos_y))
    pos_y += 28

    pygame.draw.line(display_interface.s_dir, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)

    pos_x_files = 47
    pos_y_files = 78
    files_count = 0
    for i in list(files):
        if files_count < 12:
            file_text = '{:^0.34}'.format(str(i))
            file_size_text = "Size: " + '{:^0.38}'.format(str(files[i]['Size']) + " bytes")
            file_created_text = "Access: " + '{:^0.38}'.format(str(files[i]['Access']))
            file_mod_text = "Modification: "+ '{:^0.38}'.format(str(files[i]['Mod']))

            font_file_text = display_interface.font_3.render(file_text, 10, display_interface.dark_sand)
            font_file_size_text = display_interface.font_4.render(file_size_text, 10,
display_interface.dark_sand)
            font_file_created_text = display_interface.font_4.render(file_created_text, 10,
display_interface.dark_sand)
            font_file_mod_text = display_interface.font_4.render(file_mod_text, 10,
display_interface.dark_sand)

            pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files, pos_y_files+3,
int((display_interface.WINDOW_WIDTH)/2)-75, (73)))
            pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files-5, pos_y_files-
4, int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
            pygame.draw.rect(display_interface.s_dir, display_interface.grey2, (pos_x_files-4, pos_y_files-
3, int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
            display_interface.s_dir.blit(font_file_text, (pos_x_files + 8, pos_y_files))
            pygame.draw.circle(display_interface.s_dir, display_interface.black, ((pos_x_files-
5),(pos_y_files + 8)), 8)
            pygame.draw.circle(display_interface.s_dir, display_interface.grey_blue2, ((pos_x_files-
5),(pos_y_files + 8)), 6)
            pos_y_files += 18
            pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files-5, pos_y_files,
int((display_interface.WINDOW_WIDTH)/2)-75, (53)))
            pygame.draw.rect(display_interface.s_dir, display_interface.grey2, (pos_x_files-4,
pos_y_files+1, int((display_interface.WINDOW_WIDTH)/2)-77, (51)))
            pos_y_files += 5
            display_interface.s_dir.blit(font_file_size_text, (pos_x_files, pos_y_files))
            pos_y_files += 15
            display_interface.s_dir.blit(font_file_created_text, (pos_x_files, pos_y_files))
            pos_y_files += 15
            display_interface.s_dir.blit(font_file_mod_text, (pos_x_files, pos_y_files))
            pos_y_files += 30
            files.pop(i)
```

```
            files_count += 1
            if (files_count == 6):
                pos_x_files += int(((display_interface.WINDOW_WIDTH)/2)-22)
                pos_y_files = 78
    display_interface.window.blit(display_interface.s_dir_border, (0, 0))
    display_interface.window.blit(display_interface.s_dir, (5, 5))
    return files

def files_next_page(list_left_files):
    files = list_left_files
    if len(files) > 0:
        pygame.draw.rect(display_interface.s_proc, display_interface.background,
(display_interface.WINDOW_WIDTH-110, 10, 80, 30))
        text_next_page = "Next Page"
        font_text_next_page = display_interface.font_3.render(text_next_page, 10,
display_interface.dark_sand)
        display_interface.s_dir.blit(font_text_next_page, ((display_interface.WINDOW_WIDTH-110), 10))
        pygame.draw.circle(display_interface.s_dir, display_interface.black,
((display_interface.WINDOW_WIDTH-70),40), (10 + 3))
        pygame.draw.circle(display_interface.s_dir, display_interface.dark_sand,
((display_interface.WINDOW_WIDTH-70),40), 10)
        display_interface.window.blit(display_interface.s_dir_border, (0, 0))
        display_interface.window.blit(display_interface.s_dir, (5, 5))
        return True
    elif len(files) <= 0:
        return False

def running_processes():
    processes = psutil.pids()
    process = {}
    for i in processes:
        try:
            proc_name = str(psutil.Process(i).name())
            if proc_name not in process:
                process[proc_name] = {}
                process[proc_name]['PIDs'] = []
                process[proc_name]['Status'] = []
                process[proc_name]['# Total'] = 0
            process[proc_name]['PIDs'].append(i)
            process[proc_name]['Status'].append(psutil.Process(i).status())
            process[proc_name]['# Total'] += 1
        except (psutil.AccessDenied, psutil.ZombieProcess):
            pass
        except psutil.NoSuchProcess:
            continue
    return process

def processes_page(list_process):
    process = list_process
    display_interface.s_proc.fill(display_interface.background)
    display_interface.s_proc_border.fill(display_interface.black)
    pos_x = 20
    pos_y = 10
    text_0 = "Process"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_proc.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = "Total of: " + str(len(process)) + " process"
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_proc.blit(font_text_1, (pos_x, pos_y))
```

```
    pygame.draw.line(display_interface.s_proc, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)

    pos_x_process = 47
    pos_y_process = 78
    process_count = 0
    for i in list(process):
        if process_count < 12:
            process_text = str(i)
            process_size_text = "PIDs: " + '{:^0.38}'.format(str(process[i]['PIDs']))
            process_created_text = "Status: " + '{:^0.37}'.format(str(process[i]['Status']))
            process_mod_text = "# Total: "+ '{:^0.38}'.format(str(process[i]['# Total']))

            font_process_text = display_interface.font_3.render(process_text, 10,
display_interface.dark_sand)
            font_process_size_text = display_interface.font_4.render(process_size_text, 10,
display_interface.dark_sand)
            font_process_created_text = display_interface.font_4.render(process_created_text, 10,
display_interface.dark_sand)
            font_process_mod_text = display_interface.font_4.render(process_mod_text, 10,
display_interface.dark_sand)

            pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process,
pos_y_process+3, int((display_interface.WINDOW_WIDTH)/2)-75, (73)))
            pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process-5,
pos_y_process-4, int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
            pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_process-4,
pos_y_process-3, int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
            display_interface.s_proc.blit(font_process_text, (pos_x_process + 8, pos_y_process))
            pygame.draw.circle(display_interface.s_proc, display_interface.black, ((pos_x_process-
5),(pos_y_process + 8)), 8)
            pygame.draw.circle(display_interface.s_proc, display_interface.grey_blue2, ((pos_x_process-
5),(pos_y_process + 8)), 6)
            pos_y_process += 18
            pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process-5,
pos_y_process, int((display_interface.WINDOW_WIDTH)/2)-75, (53)))
            pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_process-4,
pos_y_process+1, int((display_interface.WINDOW_WIDTH)/2)-77, (51)))
            pos_y_process += 5
            display_interface.s_proc.blit(font_process_size_text, (pos_x_process, pos_y_process))
            pos_y_process += 15
            display_interface.s_proc.blit(font_process_created_text, (pos_x_process, pos_y_process))
            pos_y_process += 15
            display_interface.s_proc.blit(font_process_mod_text, (pos_x_process, pos_y_process))
            pos_y_process += 30
            process.pop(i)

            process_count += 1
            if (process_count == 6):
                pos_x_process += int(((display_interface.WINDOW_WIDTH)/2)-22)
                pos_y_process = 78
        display_interface.window.blit(display_interface.s_proc_border, (0, 0))
        display_interface.window.blit(display_interface.s_proc, (5, 5))
    return process

def process_next_page(list_left_process):
    process = list_left_process
    if len(process) > 0:
        pygame.draw.rect(display_interface.s_proc, display_interface.background,
(display_interface.WINDOW_WIDTH-110, 10, 80, 30))
        text_next_page = "Next Page"
```

```
        font_text_next_page = display_interface.font_3.render(text_next_page, 10,
display_interface.dark_sand)
        display_interface.s_proc.blit(font_text_next_page, ((display_interface.WINDOW_WIDTH-110), 10))
        pygame.draw.circle(display_interface.s_proc, display_interface.black,
((display_interface.WINDOW_WIDTH-70),40), (10 + 3))
        pygame.draw.circle(display_interface.s_proc, display_interface.dark_sand,
((display_interface.WINDOW_WIDTH-70),40), 10)
        display_interface.window.blit(display_interface.s_proc_border, (0, 0))
        display_interface.window.blit(display_interface.s_proc, (5, 5))
        return True
    elif len(process) <= 0:
        return False
```

Código da Versão 5.0

Arquivo: main.py

```
import data_and_funcs
import display_interface

import pygame
import time
import threading

def main():
    pygame.display.set_caption("PC Resources Monitor")
    pygame.display.init()

    ### Display ###
    clock = pygame.time.Clock()
    count = 0
    close = False
    files_page_menu = '1'
    processes_page_menu = '1'
    network_page_menu = '1'
    while not close:
        #t1 = time.perf_counter()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                close = True
            ### Keyboard ###
            if event.type == pygame.KEYDOWN:
                if ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'HOME')) or ((event.key ==
pygame.K_LEFT) and (display_interface.menu == 'FILES')):
                    display_interface.menu = 'CPU'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'CPU')) or ((event.key
== pygame.K_LEFT) and (display_interface.menu == 'PROCESS')):
                    display_interface.menu = 'FILES'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'FILES')) or ((event.key
== pygame.K_LEFT) and (display_interface.menu == 'NETWORK')):
                    display_interface.menu = 'PROCESS'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'PROCESS')) or
((event.key == pygame.K_LEFT) and (display_interface.menu == 'HOME')):
                    display_interface.menu = 'NETWORK'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'NETWORK')) or
((event.key == pygame.K_LEFT) and (display_interface.menu == 'CPU')) or ((event.key == pygame.K_SPACE)):
                    display_interface.menu = 'HOME'
            ### Mouse ###
            if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                pos = pygame.mouse.get_pos()
                if display_interface.botton_display_0.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
                    display_interface.menu = 'HOME'
                elif display_interface.botton_display_1.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
                    display_interface.menu = 'CPU'
                elif display_interface.botton_display_2.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
                    display_interface.menu = 'FILES'
                elif display_interface.botton_display_3.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
                    display_interface.menu = 'PROCESS'
                elif display_interface.botton_display_4.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
                    display_interface.menu = 'NETWORK'
```

```
        count += 1
        if count == 10:
            display_interface.menu_buttons(display_interface.menu)
            if display_interface.menu == 'HOME':
                #t1 = time.perf_counter()
                data_and_funcs.cpu_graph_0()
                data_and_funcs.mem_graph_0()
                data_and_funcs.hd_graph_0()
                data_and_funcs.pc_1()
                #t2 = time.perf_counter()
                #print("Process Time on Home Page: " + str(t2 - t1))

            elif display_interface.menu == 'CPU':
                #t1 = time.perf_counter()
                CPU_thrad_1 = threading.Thread(target=data_and_funcs.cpu_graph_0)
                CPU_thrad_2 = threading.Thread(target=data_and_funcs.cpu_graph_1)
                CPU_thrad_3 = threading.Thread(target=data_and_funcs.cpu_inf)
                CPU_thrad_1.start()
                CPU_thrad_2.start()
                CPU_thrad_3.start()
                CPU_thrad_1.join()
                CPU_thrad_2.join()
                CPU_thrad_3.join()
                #t2 = time.perf_counter()
                #print("Process Time on CPU Page: " + str(t2 - t1))

            elif display_interface.menu =='FILES':
                #t1 = time.perf_counter()
                list_files = threading.Thread(target=data_and_funcs.create_files_pages)
                FILES_thrad_1 = threading.Thread(target=data_and_funcs.files_display, args=files_page_menu)
                FILES_thrad_2 = threading.Thread(target=data_and_funcs.files_page_header,
args=files_page_menu)
                list_files.start()
                FILES_thrad_1.start()
                FILES_thrad_2.start()
                list_files.join()
                FILES_thrad_1.join()
                FILES_thrad_2.join()
                if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                    pos = pygame.mouse.get_pos()
                    #print(pos)
                    if data_and_funcs.files_next_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(files_page_menu) + 1) <= data_and_funcs.files_pages_count():
                            newpage = int(files_page_menu) + 1
                            files_page_menu = str(newpage)
                            #print("Displaying Files Page:", files_page_menu)
                    if data_and_funcs.files_back_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(files_page_menu) - 1) > 0:
                            newpage = int(files_page_menu) - 1
                            files_page_menu = str(newpage)
                            #print("Displaying Files Page:", files_page_menu)
                #t2 = time.perf_counter()
                #print("Process Time on FILES Page: " + str(t2 - t1))

            elif display_interface.menu == 'PROCESS':
                #t1 = time.perf_counter()
                PROCESS_thrad_1 = threading.Thread(target=data_and_funcs.processes_display,
args=processes_page_menu)
                PROCESS_thrad_2 = threading.Thread(target=data_and_funcs.processes_page_header,
```

```
args=processes_page_menu)
                PROCESS_thrad_1.start()
                PROCESS_thrad_2.start()
                PROCESS_thrad_1.join()
                PROCESS_thrad_2.join()
                if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                    pos = pygame.mouse.get_pos()
                    #print(pos)
                    if data_and_funcs.files_next_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(processes_page_menu) + 1) <= data_and_funcs.processes_pages_number:
                            newpage = int(processes_page_menu) + 1
                            processes_page_menu = str(newpage)
                            #print("Displaying Processes Page:", processes_page_menu)
                    if data_and_funcs.files_back_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(processes_page_menu) - 1) > 0:
                            newpage = int(processes_page_menu) - 1
                            processes_page_menu = str(newpage)
                            #print("Displaying Processes Page:", processes_page_menu)
                #t2 = time.perf_counter()
                #print("Process Time on PROCESS Page: " + str(t2 - t1))

            elif display_interface.menu == 'NETWORK':
                #t1 = time.perf_counter()
                NETWORK_thrad_1 = threading.Thread(target=data_and_funcs.network_display,
args=(network_page_menu, data_and_funcs.network_pages,))
                NETWORK_thrad_2 = threading.Thread(target=data_and_funcs.network_page_header,
args=(network_page_menu, data_and_funcs.network_pages_number,))
                NETWORK_thrad_1.start()
                NETWORK_thrad_2.start()
                NETWORK_thrad_1.join()
                NETWORK_thrad_2.join()
                if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                    pos = pygame.mouse.get_pos()
                    #print(pos)
                    if data_and_funcs.files_next_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(network_page_menu) + 1) <= data_and_funcs.network_pages_number:
                            newpage = int(network_page_menu) + 1
                            network_page_menu = str(newpage)
                            #print("Displaying NETWORK Page:", network_page_menu)
                    if data_and_funcs.files_back_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(network_page_menu) - 1) > 0:
                            newpage = int(network_page_menu) - 1
                            network_page_menu = str(newpage)
                            #print("Displaying NETWORK Page:", network_page_menu)
                #t2 = time.perf_counter()
                #print("Process Time on NETWORK Page: " + str(t2 - t1))
            count = 0
        pygame.display.update()
        clock.tick(60)
    pygame.display.quit()
if __name__ == "__main__":
    main()
```

Arquivo: display_interface.py

```
import pygame
```

```
### Windows Config. ###
WINDOW_WIDTH = 800
WINDOW_HEIGHT = 600
MENU_BOTTOM_BORDER_WIDTH = WINDOW_WIDTH
MENU_BOTTOM_BORDER_HEIGHT = 40
MENU_BOTTOM_WIDTH = (MENU_BOTTOM_BORDER_WIDTH - 10)
MENU_BOTTOM_HEIGHT = (MENU_BOTTOM_BORDER_HEIGHT - 10)

window = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT + MENU_BOTTOM_BORDER_HEIGHT))

### Surfaces ###
s_menu_bottom = pygame.surface.Surface((MENU_BOTTOM_WIDTH, MENU_BOTTOM_HEIGHT))
s_menu_bottom_border = pygame.surface.Surface((MENU_BOTTOM_BORDER_WIDTH, MENU_BOTTOM_BORDER_HEIGHT))
s_display = pygame.surface.Surface((int(MENU_BOTTOM_WIDTH / 2), MENU_BOTTOM_HEIGHT))
s = pygame.surface.Surface((WINDOW_WIDTH, int((WINDOW_HEIGHT)/4)))
s_dir = pygame.surface.Surface((WINDOW_WIDTH-10, WINDOW_HEIGHT-5))
s_dir_border = pygame.surface.Surface((WINDOW_WIDTH, WINDOW_HEIGHT))
s_header = pygame.surface.Surface((WINDOW_WIDTH-10, 65))
s_proc = pygame.surface.Surface((WINDOW_WIDTH-10, WINDOW_HEIGHT-5))
s_proc_border = pygame.surface.Surface((WINDOW_WIDTH, WINDOW_HEIGHT))
s_info = pygame.surface.Surface(((WINDOW_WIDTH - 10), int((WINDOW_HEIGHT)/4) - 5))
s_graph_1 = pygame.surface.Surface((WINDOW_WIDTH, int(((WINDOW_HEIGHT)/4)*2)))

### Colors ###
red = (150,0,0)
green = (255,0,0)
blue = (0,0,255)
light_blue = (0,200,255)
black = (0,0,0)
white = (255, 255, 255)
grey = (150, 150, 150)
grey1 = (80, 80, 80)
grey2 = (45, 45, 45)
grey2 = (55, 55, 55)
grey_blue1 = (156, 181, 201)
grey_blue2 = (0, 55, 130)
dark_blue = (0, 0, 50)
sand = (255, 230, 185)
sand2 = (225, 190, 145)
dark_sand = (165, 145, 110)
background = (45, 45, 45)

### Fonts ###
pygame.font.init()
def make_font(fonts, size):
    available = pygame.font.get_fonts()
    # get_fonts() returns a list of lowercase spaceless font names
    choices = map(lambda x:x.lower().replace(' ', ''), fonts)
    for choice in choices:
        if choice in available:
            return pygame.font.SysFont(choice, size)
    return pygame.font.Font(None, size)
_cached_fonts = {}
def get_font(font_preferences, size):
    global _cached_fonts
    key = str(font_preferences) + '|' + str(size)
    font = _cached_fonts.get(key, None)
    if font == None:
        font = make_font(font_preferences, size)
        _cached_fonts[key] = font
```

```
    return font

font_preferences = ["Consolas","Rockwell", "Calibri", "Arial", ]

font_1 = get_font(font_preferences, 30)
font_2 = get_font(font_preferences, 20)
font_3 = get_font(font_preferences, 16)
font_4 = get_font(font_preferences, 12)

### Menu ###
### Menu Settings ###
pos_x = 125
pos_y = (int(MENU_BOTTOM_BORDER_HEIGHT / 3) + 2)
button_size_pressed = 8
button_size_nonpressed = 10

pos_x_s_display = 175
pos_y_s_display = WINDOW_HEIGHT + 5

def menu_buttons(menu):
    s_menu_bottom_border.fill(black)
    s_menu_bottom.fill(background)
    s_display.fill(background)

    #botton HOME
    pygame.draw.circle(s_display, black, (pos_x,pos_y), (button_size_nonpressed + 3))
    botton_display_0 = pygame.draw.circle(s_display, dark_sand, (pos_x,pos_y), button_size_nonpressed)
    #botton CPU
    pygame.draw.circle(s_display, black, ((pos_x + 50),pos_y), (button_size_nonpressed + 3))
    botton_display_1 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 50),pos_y),
button_size_nonpressed)
    #botton FILES
    pygame.draw.circle(s_display, black, ((pos_x + 100),pos_y), (button_size_nonpressed + 3))
    botton_display_2 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 100),pos_y),
button_size_nonpressed)
    #botton PROCESS
    pygame.draw.circle(s_display, black, ((pos_x + 150),pos_y), (button_size_nonpressed + 3))
    botton_display_3 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 150),pos_y),
button_size_nonpressed)
    #botton NETWORK
    pygame.draw.circle(s_display, black, ((pos_x + 200),pos_y), (button_size_nonpressed + 3))
    botton_display_4 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 200),pos_y),
button_size_nonpressed)

    if menu == 'HOME':
        pygame.draw.circle(s_display, black, (pos_x,pos_y), (button_size_pressed + 3))
        botton_display_0 = pygame.draw.circle(s_display, grey1, (pos_x,pos_y), button_size_pressed)
    elif menu == 'CPU':
        pygame.draw.circle(s_display, black, ((pos_x + 50),pos_y), (button_size_pressed + 3))
        botton_display_1 = pygame.draw.circle(s_display, grey1, ((pos_x + 50),pos_y), button_size_pressed)
    elif menu =='FILES':
        pygame.draw.circle(s_display, black, ((pos_x + 100),pos_y), (button_size_pressed + 3))
        botton_display_2 = pygame.draw.circle(s_display, grey1, ((pos_x + 100),pos_y), button_size_pressed)
    elif menu == 'PROCESS':
        pygame.draw.circle(s_display, black, ((pos_x + 150),pos_y), (button_size_pressed + 3))
        botton_display_3 = pygame.draw.circle(s_display, grey1, ((pos_x + 150),pos_y), button_size_pressed)
    elif menu == 'NETWORK':
        pygame.draw.circle(s_display, black, ((pos_x + 200),pos_y), (button_size_pressed + 3))
        botton_display_4 = pygame.draw.circle(s_display, grey1, ((pos_x + 200),pos_y), button_size_pressed)

    pages = font_4.render("Tabs", 10, dark_sand)
```

```
    s_menu_bottom.blit(pages, (20, 10))
    ass = font_4.render("Dev.: Fábio R. P. Nunes", 10, dark_sand)
    s_menu_bottom.blit(ass, (610, 10))
    window.blit(s_menu_bottom_border, (0, WINDOW_HEIGHT))
    window.blit(s_menu_bottom, (5, (WINDOW_HEIGHT + 5)))
    window.blit(s_display, (pos_x_s_display, pos_y_s_display))

    return botton_display_0, botton_display_1, botton_display_2, botton_display_3, botton_display_4

menu = 'HOME'
botton_display_0 = menu_buttons(menu)[0]
botton_display_1 = menu_buttons(menu)[1]
botton_display_2 = menu_buttons(menu)[2]
botton_display_3 = menu_buttons(menu)[3]
botton_display_4 = menu_buttons(menu)[4]
```

Arquivo: data_and_funcs.py

```
import display_interface

import pygame
import psutil
import cpuinfo
import platform
import subprocess
import threading
import socket
import os
import datetime
import nmap

### Static Data ###
# PC Inf 1
pc_name = platform.node()
operational_system = platform.system()
osVersion = platform.version()
arc = platform.machine()
processor = platform.processor()
cores = psutil.cpu_count(logical=True)
pc_partitions = psutil.disk_partitions(all)
primary_storage = pc_partitions[0][0]
#local = os.path.dirname(os.path.realpath(__file__))
local = 'D:\\Coding\\Python\\Projeto de Bloco\\Tests'

# IPs
#hosts_mapping_ready = True
#network_mapping_ready = True
#IPs_mapping_ready = True
def get_ip_addresses(family):
    for interface, snics in psutil.net_if_addrs().items():
        for snic in snics:
            if snic.family == family:
                yield (interface, snic.address)

def ping(hostname):
    args = []

    if operational_system == "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 host(subnetwork):
    print("Mapping", subnetwork, "subnetwork...\r")
    hosts_list = []
    return_codes = dict()
    for i in range(1, 255):
        host_test = (subnetwork + '{0}'.format(i))
        return_codes[subnetwork + '{0}'.format(i)] = ping(host_test)
        if i %20 == 0:
            print(".", end = "")
        if return_codes[subnetwork + '{0}'.format(i)] == 0:
            hosts_list.append(subnetwork + '{0}'.format(i))
    print("\nMapping ready...")

    hosts_mapping_ready = True
    print("hosts_mapping_ready:", hosts_mapping_ready)
    return hosts_list

def hosts_data(hosts_list):
    nm = nmap.PortScanner()
    IPs = dict()
    for i in hosts_list:
        print("Scanning", i, "IP...")
        nm.scan(i)
        IPs[i] = nm[i]

    IPs_Scanned = True
    IPs_mapping_ready = True
    return IPs

ipv4s = list(get_ip_addresses(socket.AF_INET))
ipv6s = list(get_ip_addresses(socket.AF_INET6))
mac_address = list(get_ip_addresses(psutil.AF_LINK))
subnetwork = ".".join(ipv4s[0][1].split('.')[0:3]) + '.'

hosts_scanned = host(subnetwork)
#hosts_scanned = ['192.168.0.1', '192.168.0.17', '192.168.0.32', '192.168.0.50']

IPs_Info = hosts_data(hosts_scanned)
#IPs_Info = {'192.168.0.1': {'hostnames': [{'name': '', 'type': ''}], 'addresses': {'ipv4': '192.168.0.1',
'mac': '8C:44:4F:73:C0:D1'}, 'vendor': {'8C:44:4F:73:C0:D1': 'Humax'}, 'status': {'state': 'up', 'reason':
'arp-response'}, 'tcp': {22: {'state': 'filtered', 'reason': 'no-response', 'name': 'ssh', 'product': '',
'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 80: {'state': 'open', 'reason': 'syn-ack', 'name':
'http', 'product': 'micro_httpd', 'version': '', 'extrainfo': '', 'conf': '10', 'cpe':
'cpe:/a:acme:micro_httpd'}, 139: {'state': 'filtered', 'reason': 'no-response', 'name': 'netbios-ssn',
'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 5431: {'state': 'open', 'reason':
'syn-ack', 'name': 'upnp', 'product': 'Belkin/Linksys wireless router UPnP', 'version': '', 'extrainfo':
'UPnP 1.0; BRCM400 1.0', 'conf': '10', 'cpe': 'cpe:/o:linux:linux_kernel:2.4'}, 12345: {'state':
'filtered', 'reason': 'no-response', 'name': 'netbus', 'product': '', 'version': '', 'extrainfo': '',
'conf': '3', 'cpe': ''}}}, '192.168.0.17': {'hostnames': [{'name': '', 'type': ''}], 'addresses': {'ipv4':
'192.168.0.17', 'mac': '1A:32:21:02:17:E1'}, 'vendor': {}, 'status': {'state': 'up', 'reason': 'arp-
response'}}, '192.168.0.32': {'hostnames': [{'name': '', 'type': ''}], 'addresses': {'ipv4':
'192.168.0.32'}, 'vendor': {}, 'status': {'state': 'up', 'reason': 'localhost-response'}, 'tcp': {135:
{'state': 'open', 'reason': 'syn-ack', 'name': 'msrpc', 'product': 'Microsoft Windows RPC', 'version': '',
'extrainfo': '', 'conf': '10', 'cpe': 'cpe:/o:microsoft:windows'}, 139: {'state': 'open', 'reason': 'syn-
```

```
ack', 'name': 'netbios-ssn', 'product': 'Microsoft Windows netbios-ssn', 'version': '', 'extrainfo': '',
'conf': '10', 'cpe': 'cpe:/o:microsoft:windows'}, 445: {'state': 'open', 'reason': 'syn-ack', 'name':
'microsoft-ds', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}}}, '192.168.0.50':
{'hostnames': [{'name': '', 'type': ''}], 'addresses': {'ipv4': '192.168.0.50', 'mac':
'54:B8:0A:9C:DF:E8'}, 'vendor': {'54:B8:0A:9C:DF:E8': 'D-Link International'}, 'status': {'state': 'up',
'reason': 'arp-response'}, 'tcp': {23: {'state': 'filtered', 'reason': 'no-response', 'name': 'telnet',
'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 80: {'state': 'open', 'reason':
'syn-ack', 'name': 'http', 'product': 'mini_httpd', 'version': '1.19 19dec2003', 'extrainfo': '', 'conf':
'10', 'cpe': 'cpe:/a:acme:mini_httpd:1.19_19dec2003'}, 139: {'state': 'open', 'reason': 'syn-ack', 'name':
'tcpwrapped', 'product': '', 'version': '', 'extrainfo': '', 'conf': '8', 'cpe': ''}, 1050: {'state':
'open', 'reason': 'syn-ack', 'name': 'http', 'product': 'mini_httpd', 'version': '1.19 19dec2003',
'extrainfo': '', 'conf': '10', 'cpe': 'cpe:/a:acme:mini_httpd:1.19_19dec2003'}}}}
#IPs_Info = None

### Dynamic Data ###
def cpu_graph_0():
    ### DATA ###
    cpu = int(psutil.cpu_percent(interval=0.1))
    freq_per_cent = int(psutil.cpu_percent(interval=0.1))
    freq_max = psutil.cpu_freq().max
    freq_current = ((freq_max * freq_per_cent)/100)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*cpu) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,54), (790, 54), 6)

    bar_text = "CPU Usage"
    bar_text_2 = "- Current Frequency (Mhz): " + str(freq_current)
    bar_text_inside_01 = str(cpu) + "%"
    bar_text_inside_02 = str((100 - cpu)) + "%"


    text = display_interface.font_1.render(bar_text, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)

    display_interface.s.blit(text, (20, 18))
    display_interface.s.blit(text_2, (185, 26))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (700, 110))

    display_interface.window.blit(display_interface.s, (0, 0))

def cpu_graph_1():
    ### Data ###
    cores_percent = psutil.cpu_percent(interval=1, percpu=True)

    display_interface.s_graph_1.fill(display_interface.background)

    cores = len(cores_percent)
```

```
    x = y = shift = 10

    height = int(display_interface.s_graph_1.get_height() - 2*y)
    width = int((display_interface.s_graph_1.get_width() - 2*y - (cores + 1)*shift ) / cores)

    d = x + shift
    cores_count = 1
    for i in cores_percent:
        pygame.draw.rect(display_interface.s_graph_1, display_interface.black, (d+5, y+5, width, height))
        pygame.draw.rect(display_interface.s_graph_1, display_interface.red, (d, y, width, height))
        pygame.draw.rect(display_interface.s_graph_1, display_interface.grey_blue2, (d, y, width, ((1-
i/100)*height)))
        bar_text_inside_0 = str(int(i)) + "%"
        text_inside_0 = display_interface.font_3.render(bar_text_inside_0, 10, display_interface.sand2)
        display_interface.s_graph_1.blit(text_inside_0, ((d + (width/3)), (height - 10)))
        bar_text_inside_1 = "#"+str(cores_count)
        text_inside_1 = display_interface.font_3.render(bar_text_inside_1, 10, display_interface.dark_sand)
        display_interface.s_graph_1.blit(text_inside_1, ((d + (width/3)), y+10))
        cores_count += 1
        d = d + width + shift

    display_interface.window.blit(display_interface.s_graph_1, (0, (display_interface.WINDOW_HEIGHT)/4))

def mem_graph_0():
    ### DATA ###
    mem = psutil.virtual_memory()
    memTotalGigas = round((mem.total / (1024*1024*1024)), 2)
    memUsedGigas = round((mem.used / (1024*1024*1024)), 2)
    memFreeGigas = round((mem.free / (1024*1024*1024)), 2)
    mem_per_cent = int(mem.percent)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*mem.percent) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH - 10), 45), 6)

    bar_text_1 = "Memory Usage"
    bar_text_2 = "(Total: " + str(memTotalGigas) + " GB)"
    bar_text_inside_01 = str(memUsedGigas) + " Used"
    bar_text_inside_02 = str(memFreeGigas) + " Free"
    bar_text_inside_03 = str(mem_per_cent) + "%"
    bar_text_inside_04 = str(100 - mem_per_cent) + "%"

    text_1 = display_interface.font_1.render(bar_text_1, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)
    text_inside3 = display_interface.font_2.render(bar_text_inside_03, 10, display_interface.sand2)
    text_inside4 = display_interface.font_2.render(bar_text_inside_04, 10, display_interface.dark_sand)

    display_interface.s.blit(text_1, (20, 10))
    display_interface.s.blit(text_2, (235, 18))
```

```
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (655, 110))
    display_interface.s.blit(text_inside3, (35, 75))
    display_interface.s.blit(text_inside4, (730, 75))

    display_interface.window.blit(display_interface.s, (0, ((display_interface.WINDOW_HEIGHT)/4)))

def hd_graph_0():
    ### DATA ###
    hd = psutil.disk_usage(str(primary_storage))
    hdTotalGigas = round((hd.total / (1024*1024*1024)), 2)
    hdUsedGigas = round((hd.used / (1024*1024*1024)), 2)
    hdFreeGigas = round((hd.free / (1024*1024*1024)), 2)
    hd_per_cent = int(hd.percent)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*hd.percent) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,45), (790, 45), 6)

    bar_text_1 = "Primary Storage Usage"
    bar_text_2 = "(Total: " + str(hdTotalGigas) + " GB)"
    bar_text_inside_01 = str(hdUsedGigas) + " Used"
    bar_text_inside_02 = str(hdFreeGigas) + " Free"
    bar_text_inside_03 = str(hd_per_cent) + "%"
    bar_text_inside_04 = str(100 - hd_per_cent) + "%"

    text_1 = display_interface.font_1.render(bar_text_1, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)
    text_inside3 = display_interface.font_2.render(bar_text_inside_03, 10, display_interface.sand2)
    text_inside4 = display_interface.font_2.render(bar_text_inside_04, 10, display_interface.dark_sand)

    display_interface.s.blit(text_1, (20, 10))
    display_interface.s.blit(text_2, (390, 18))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (640, 110))
    display_interface.s.blit(text_inside3, (35, 75))
    display_interface.s.blit(text_inside4, (730, 75))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*2))))

def pc_1():
    display_interface.s_info.fill(display_interface.background)
    display_interface.s.fill(display_interface.black)

    pygame.draw.line(display_interface.s_info, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH -20), 45), 6)
    pos_y = 10
    text_0 = "PC Information"
    text_1 = "PC Name: " + str(pc_name)
    text_2 = "Operating System: " + str(operational_system) + " " + str(osVersion)
```

```
    text_3 = "Networking Information"

    ipv4s_text_pos_y = 85
    text_4 = "- IPv4s"
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)
    display_interface.s_info.blit(font_text_4, (20, ipv4s_text_pos_y))
    ipv4s_text_pos_y += 15
    for i in range(len(ipv4s)):
        text = ipv4s[i][0] + ": " + ipv4s[i][1]
        font_text = display_interface.font_4.render(text, 10, display_interface.dark_sand)
        display_interface.s_info.blit(font_text, (20, ipv4s_text_pos_y))
        ipv4s_text_pos_y += 15

    text_4 = "MAC Address: " + str(mac_address[0][1])

    ipv6s_text_pos_y = 55
    text_5 = "- IPv6s"
    font_text_5 = display_interface.font_4.render(text_5, 10, display_interface.dark_sand)
    display_interface.s_info.blit(font_text_5, (400, ipv6s_text_pos_y))
    ipv6s_text_pos_y += 15
    for i in range(len(ipv6s)):
        text = ipv6s[i][0] + ": " + ipv6s[i][1]
        font_text = display_interface.font_4.render(text, 10, display_interface.dark_sand)
        display_interface.s_info.blit(font_text, (400, ipv6s_text_pos_y))
        ipv6s_text_pos_y += 15

    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    font_text_1 = display_interface.font_4.render(text_1, 10, display_interface.grey)
    font_text_2 = display_interface.font_4.render(text_2, 10, display_interface.grey)
    font_text_3 = display_interface.font_4.render(text_3, 10, display_interface.dark_sand)
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)

    display_interface.s_info.blit(font_text_0, (20, pos_y))
    display_interface.s_info.blit(font_text_1, (275, pos_y))
    pos_y += 15
    display_interface.s_info.blit(font_text_2, (275, pos_y))
    pos_y += 30
    display_interface.s_info.blit(font_text_3, (20, pos_y))
    display_interface.s_info.blit(font_text_4, (20, 70))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*3))))
    display_interface.window.blit(display_interface.s_info, (5,
(((((display_interface.WINDOW_HEIGHT)/4)*3)+5))))

def cpu_inf():
    inf = cpuinfo.get_cpu_info()
    cpu_name = inf['brand_raw']
    cpu_arch = inf['arch']
    cpu_bits = inf['bits']

    p_cores = psutil.cpu_count(logical=False)
    l_cores = psutil.cpu_count()

    display_interface.s_info.fill(display_interface.background)
    display_interface.s.fill(display_interface.black)

    pygame.draw.line(display_interface.s_info, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH -20), 45), 6)

    text_0 = "CPU Information"
    text_1 = "CPU Name: " + str(cpu_name)
```

```
    text_2 = "Architecture: " + str(cpu_arch) + " (" + str(arc) + ")"
    text_3 = "Word Length: " + str(cpu_bits) + " bits"
    text_4 = "Physical Cores: " + str(p_cores)
    text_5 = "Logical Cores: " + str(l_cores)

    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    font_text_1 = display_interface.font_4.render(text_1, 10, display_interface.dark_sand)
    font_text_2 = display_interface.font_4.render(text_2, 10, display_interface.dark_sand)
    font_text_3 = display_interface.font_4.render(text_3, 10, display_interface.dark_sand)
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)
    font_text_5 = display_interface.font_4.render(text_5, 10, display_interface.dark_sand)

    display_interface.s_info.blit(font_text_0, (20, 10))
    display_interface.s_info.blit(font_text_1, (20, 55))
    display_interface.s_info.blit(font_text_2, (20, 70))
    display_interface.s_info.blit(font_text_3, (20, 85))
    display_interface.s_info.blit(font_text_4, (20, 100))
    display_interface.s_info.blit(font_text_5, (20, 115))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*3))))
    display_interface.window.blit(display_interface.s_info, (5,
(((((display_interface.WINDOW_HEIGHT)/4)*3)+5))))

### Files & Processes Page
pos_x_s_header = 5
pos_y_s_header = 5

def display_next_page_buttom():
    text_next_page = "Next"
    #pygame.draw.circle(s_header, black, ((WINDOW_WIDTH-40),40), (10 + 3))
    files_next_page_buttom = pygame.draw.circle(display_interface.s_header, display_interface.background,
((display_interface.WINDOW_WIDTH-50),40), 20)
    pygame.draw.polygon(display_interface.s_header, display_interface.black,
((display_interface.WINDOW_WIDTH-35,45),(display_interface.WINDOW_WIDTH-
55,35),(display_interface.WINDOW_WIDTH-55,55)))
    pygame.draw.polygon(display_interface.s_header, display_interface.dark_sand,
((display_interface.WINDOW_WIDTH-40,40),(display_interface.WINDOW_WIDTH-
60,30),(display_interface.WINDOW_WIDTH-60,50)))
    font_text_next_page = display_interface.font_3.render(text_next_page, 10, display_interface.dark_sand)
    display_interface.s_header.blit(font_text_next_page, ((display_interface.WINDOW_WIDTH-70), 10))
    return files_next_page_buttom

def display_back_page_buttom():
    text_next_page = "Back"
    #pygame.draw.circle(s_header, black, ((WINDOW_WIDTH-100),40), (10 + 3))
    files_back_page_buttom = pygame.draw.circle(display_interface.s_header, display_interface.background,
((display_interface.WINDOW_WIDTH-100),40), 20)
    pygame.draw.polygon(display_interface.s_header, display_interface.black,
((display_interface.WINDOW_WIDTH-110,45),(display_interface.WINDOW_WIDTH-
90,35),(display_interface.WINDOW_WIDTH-90,55)))
    pygame.draw.polygon(display_interface.s_header, display_interface.dark_sand,
((display_interface.WINDOW_WIDTH-115,40),(display_interface.WINDOW_WIDTH-
95,30),(display_interface.WINDOW_WIDTH-95,50)))
    font_text_next_page = display_interface.font_3.render(text_next_page, 10, display_interface.dark_sand)
    display_interface.s_header.blit(font_text_next_page, ((display_interface.WINDOW_WIDTH-120), 10))
    return files_back_page_buttom
files_next_page_buttom = display_next_page_buttom()
files_back_page_buttom = display_back_page_buttom()

## Files Page
def files_page_header(files_page_menu):
```

```
    display_interface.s_header.fill(display_interface.background)

    if (((files_pages_count()) > (1)) and ((int(files_page_menu)) < (files_pages_count()))):
        display_next_page_buttom()
    if files_page_menu != '1':
        display_back_page_buttom()

    pos_x = 20
    pos_y = 10
    text_0 = "Local Files"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = str(local)
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_1, (pos_x, pos_y))
    pos_y += 28

    pygame.draw.line(display_interface.s_header, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)

    display_interface.window.blit(display_interface.s_header, (pos_x_s_header, pos_y_s_header))

def dir_files():
    lista = os.listdir(local)
    files = {}
    for i in lista:
        filepath = os.path.join(local, i)
        if os.path.isfile(filepath):
            files[i] = {}
            files[i]['Size'] = os.stat(filepath).st_size
            files[i]['Created'] = datetime.datetime.fromtimestamp(os.stat(filepath).st_ctime)
            files[i]['Mod'] = datetime.datetime.fromtimestamp(os.stat(filepath).st_mtime)
    return files

def create_files_pages():
    files = dir_files()
    files_count = 1
    pages_count = 1
    pages = {}
    pages[pages_count] = {}
    for i in list(files):
        pages[pages_count][i] = {}
        pages[pages_count][i]['Size'] = files[i]['Size']
        pages[pages_count][i]['Created'] = files[i]['Created']
        pages[pages_count][i]['Mod'] = files[i]['Mod']
        files_count += 1
        if files_count > 12:
            pages_count += 1
            pages[pages_count] = {}
            files_count = 1
    return pages

def files_pages_count():
    return len(create_files_pages())

def files_display(files_page_menu):
    display_interface.s_dir.fill(display_interface.background)
    display_interface.s_dir_border.fill(display_interface.black)

    files = create_files_pages()[int(files_page_menu)]
```

```
    pos_x_files = 47
    pos_y_files = 78
    files_count = 0
    for i in list(files):
        file_text = '{:^0.34}'.format(str(i))
        file_size_text = "Size: " + '{:^0.38}'.format(str(files[i]['Size']) + " bytes")
        file_created_text = "Created: " + '{:^0.38}'.format(str(files[i]['Created']))
        file_mod_text = "Modification: "+ '{:^0.38}'.format(str(files[i]['Mod']))

        font_file_text = display_interface.font_3.render(file_text, 10, display_interface.dark_sand)
        font_file_size_text = display_interface.font_4.render(file_size_text, 10,
display_interface.dark_sand)
        font_file_created_text = display_interface.font_4.render(file_created_text, 10,
display_interface.dark_sand)
        font_file_mod_text = display_interface.font_4.render(file_mod_text, 10,
display_interface.dark_sand)

        pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files, pos_y_files+3,
int((display_interface.WINDOW_WIDTH)/2)-75, (73)))
        pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files-5, pos_y_files-4,
int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
        pygame.draw.rect(display_interface.s_dir, display_interface.grey2, (pos_x_files-4, pos_y_files-3,
int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
        display_interface.s_dir.blit(font_file_text, (pos_x_files + 8, pos_y_files))
        pygame.draw.circle(display_interface.s_dir, display_interface.black, ((pos_x_files-5),(pos_y_files
+ 8)), 8)
        pygame.draw.circle(display_interface.s_dir, display_interface.grey_blue2, ((pos_x_files-
5),(pos_y_files + 8)), 6)
        pos_y_files += 18
        pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files-5, pos_y_files,
int((display_interface.WINDOW_WIDTH)/2)-75, (53)))
        pygame.draw.rect(display_interface.s_dir, display_interface.grey2, (pos_x_files-4, pos_y_files+1,
int((display_interface.WINDOW_WIDTH)/2)-77, (51)))
        pos_y_files += 5
        display_interface.s_dir.blit(font_file_size_text, (pos_x_files, pos_y_files))
        pos_y_files += 15
        display_interface.s_dir.blit(font_file_created_text, (pos_x_files, pos_y_files))
        pos_y_files += 15
        display_interface.s_dir.blit(font_file_mod_text, (pos_x_files, pos_y_files))
        pos_y_files += 30

        files_count += 1
        if (files_count == 6):
            pos_x_files += int(((display_interface.WINDOW_WIDTH)/2)-22)
            pos_y_files = 78

    display_interface.window.blit(display_interface.s_dir_border, (0, 0))
    display_interface.window.blit(display_interface.s_dir, (5, 5))

### Processes Page
def processes_page_header(processes_page_menu):
    display_interface.s_header.fill(display_interface.background)

    if (((processes_pages_number) > (1)) and ((int(processes_page_menu)) < (processes_pages_number))):
        display_next_page_buttom()
    if processes_page_menu != '1':
        display_back_page_buttom()

    pos_x = 20
    pos_y = 10
```

```
    text_0 = "Process"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = "Running a total of: " + str(len(runned_process)) + " process"
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_1, (pos_x, pos_y))
    pos_y += 28

    pygame.draw.line(display_interface.s_header, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)

    display_interface.window.blit(display_interface.s_header, (pos_x_s_header, pos_y_s_header))

def running_processes():
    processes = psutil.pids()
    process = {}
    for i in processes:
        try:
            proc_name = str(psutil.Process(i).name())
            if proc_name not in process:
                process[proc_name] = {}
                process[proc_name]['PIDs'] = []
                process[proc_name]['Status'] = []
                process[proc_name]['# Total'] = 0
            process[proc_name]['PIDs'].append(i)
            process[proc_name]['Status'].append(psutil.Process(i).status())
            process[proc_name]['# Total'] += 1
        except (psutil.AccessDenied, psutil.ZombieProcess, psutil.NoSuchProcess):
            continue
    return process
runned_process = running_processes()

def create_processes_pages():
    process = runned_process
    process_count = 1
    pages_count = 1
    pages = {}
    pages[pages_count] = {}
    for i in list(process):
        pages[pages_count][i] = {}
        pages[pages_count][i]['PIDs'] = process[i]['PIDs']
        pages[pages_count][i]['Status'] = process[i]['Status']
        pages[pages_count][i]['# Total'] = process[i]['# Total']
        process_count += 1
        if process_count > 12:
            pages_count += 1
            pages[pages_count] = {}
            process_count = 1
    return pages
processes_pages = create_processes_pages()

def processes_pages_count():
    return len(processes_pages)
processes_pages_number = processes_pages_count()

def processes_display(processes_page_menu):
    display_interface.s_proc.fill(display_interface.background)
    display_interface.s_proc_border.fill(display_interface.black)

    process = processes_pages[int(processes_page_menu)]
```

```
    pos_x_process = 47
    pos_y_process = 78
    process_count = 0
    for i in list(process):
        process_text = str(i)
        process_size_text = "PIDs: " + '{:^0.38}'.format(str(process[i]['PIDs']))
        process_created_text = "Status: " + '{:^0.37}'.format(str(process[i]['Status']))
        process_mod_text = "# Total: "+ '{:^0.38}'.format(str(process[i]['# Total']))

        font_process_text = display_interface.font_3.render(process_text, 10, display_interface.dark_sand)
        font_process_size_text = display_interface.font_4.render(process_size_text, 10,
display_interface.dark_sand)
        font_process_created_text = display_interface.font_4.render(process_created_text, 10,
display_interface.dark_sand)
        font_process_mod_text = display_interface.font_4.render(process_mod_text, 10,
display_interface.dark_sand)

        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process,
pos_y_process+3, int((display_interface.WINDOW_WIDTH)/2)-75, (73)))
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process-5,
pos_y_process-4, int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_process-4,
pos_y_process-3, int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
        display_interface.s_proc.blit(font_process_text, (pos_x_process + 8, pos_y_process))
        pygame.draw.circle(display_interface.s_proc, display_interface.black, ((pos_x_process-
5),(pos_y_process + 8)), 8)
        pygame.draw.circle(display_interface.s_proc, display_interface.grey_blue2, ((pos_x_process-
5),(pos_y_process + 8)), 6)
        pos_y_process += 18
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process-5,
pos_y_process, int((display_interface.WINDOW_WIDTH)/2)-75, (53)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_process-4,
pos_y_process+1, int((display_interface.WINDOW_WIDTH)/2)-77, (51)))
        pos_y_process += 5
        display_interface.s_proc.blit(font_process_size_text, (pos_x_process, pos_y_process))
        pos_y_process += 15
        display_interface.s_proc.blit(font_process_created_text, (pos_x_process, pos_y_process))
        pos_y_process += 15
        display_interface.s_proc.blit(font_process_mod_text, (pos_x_process, pos_y_process))
        pos_y_process += 30

        process_count += 1
        if (process_count == 6):
            pos_x_process += int(((display_interface.WINDOW_WIDTH)/2)-22)
            pos_y_process = 78

        display_interface.window.blit(display_interface.s_proc_border, (0, 0))
        display_interface.window.blit(display_interface.s_proc, (5, 5))

### Network Page
def network_page_header(network_page_menu, network_pages_number):
    display_interface.s_header.fill(display_interface.background)

    if (((network_pages_number) > (1)) and ((int(network_page_menu)) < (network_pages_number))):
        display_next_page_buttom()
    if network_page_menu != '1':
        display_back_page_buttom()

    pos_x = 20
    pos_y = 10
```

```
    text_0 = "Networking Information"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = "Total of: " + str(((len(hosts_scanned)) - 1)) + " Others Local IPs on " + subnetwork + "
Subnetwork"
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_1, (pos_x, pos_y))
    pos_y += 28

    pygame.draw.line(display_interface.s_header, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)

    display_interface.window.blit(display_interface.s_header, (pos_x_s_header, pos_y_s_header))

def hosts_info():
    IPs = dict()
    for i in hosts_scanned:
        IPs[i] = dict()
        try:
            IPs[i]['mac'] = IPs_Info[i]['addresses']['mac']
        except:
            pass
        try:
            IPs[i]['vendor'] = IPs_Info[i]['vendor'][IPs_Info[i]['addresses']['mac']]
        except:
            pass
        try:
            IPs[i]['response'] = IPs_Info[i]['status']['reason']
        except:
            pass
        try:
            IPs[i]['product'] = IPs_Info[i]['product']
        except:
            pass
    return IPs
IPs_data = hosts_info()

def create_network_pages(IPs_data):
    IPs = IPs_data
    IPs_count = 1
    pages_count = 1
    pages = dict()
    pages[pages_count] = dict()
    for i in list(IPs):
        if IPs[i]['response'] != 'localhost-response':
            pages[pages_count][i] = dict()
            try:
                pages[pages_count][i]['mac'] = IPs[i]['mac']
            except:
                pass
            try:
                pages[pages_count][i]['vendor'] = IPs[i]['vendor']
            except:
                pass
            try:
                pages[pages_count][i]['response'] = IPs[i]['response']
            except:
                pass

        IPs_count += 1
```

```
        if IPs_count > 12:
            pages_count += 1
            pages[pages_count] = {}
            IPs_count = 1
    return pages
network_pages = create_network_pages(IPs_data)

def network_pages_count(network_pages):
    return len(network_pages)
network_pages_number = network_pages_count(network_pages)

def network_display(network_page_menu, network_pages):
    display_interface.s_proc.fill(display_interface.background)
    display_interface.s_proc_border.fill(display_interface.black)

    pos_x_IPs = 47
    pos_y_IPs = 78
    IPs_count = 0

    IPs = network_pages[int(network_page_menu)]

    for i in list(IPs):
        IPs_text = str(i)
        try:
            IPs_size_text = "MAC: " + '{:^0.38}'.format(str(IPs[i]['mac']))
        except:
            pass
        try:
            IPs_created_text = "Vendor: " + '{:^0.37}'.format(str(IPs[i]['vendor']))
        except:
            pass
        try:
            IPs_mod_text = "Rensponse: "+ '{:^0.38}'.format(str(IPs[i]['response']))
        except:
            pass

        font_IPs_text = display_interface.font_3.render(IPs_text, 10, display_interface.dark_sand)
        font_IPs_size_text = display_interface.font_4.render(IPs_size_text, 10,
display_interface.dark_sand)
        font_IPs_created_text = display_interface.font_4.render(IPs_created_text, 10,
display_interface.dark_sand)
        font_IPs_mod_text = display_interface.font_4.render(IPs_mod_text, 10, display_interface.dark_sand)

        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_IPs, pos_y_IPs+3,
int((display_interface.WINDOW_WIDTH)/2)-75, (73)))
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_IPs-5, pos_y_IPs-4,
int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_IPs-4, pos_y_IPs-3,
int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
        display_interface.s_proc.blit(font_IPs_text, (pos_x_IPs + 8, pos_y_IPs))
        pygame.draw.circle(display_interface.s_proc, display_interface.black, ((pos_x_IPs-5),(pos_y_IPs +
8)), 8)
        pygame.draw.circle(display_interface.s_proc, display_interface.grey_blue2, ((pos_x_IPs-
5),(pos_y_IPs + 8)), 6)
        pos_y_IPs += 18
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_IPs-5, pos_y_IPs,
int((display_interface.WINDOW_WIDTH)/2)-75, (53)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_IPs-4, pos_y_IPs+1,
int((display_interface.WINDOW_WIDTH)/2)-77, (51)))
        pos_y_IPs += 5
        display_interface.s_proc.blit(font_IPs_size_text, (pos_x_IPs, pos_y_IPs))
```

```
        pos_y_IPs += 15
        display_interface.s_proc.blit(font_IPs_created_text, (pos_x_IPs, pos_y_IPs))
        pos_y_IPs += 15
        display_interface.s_proc.blit(font_IPs_mod_text, (pos_x_IPs, pos_y_IPs))
        pos_y_IPs += 30

        IPs_count += 1
        if (IPs_count == 6):
            pos_x_IPs += int(((display_interface.WINDOW_WIDTH)/2)-22)
            pos_y_IPs = 78

    display_interface.window.blit(display_interface.s_proc_border, (0, 0))
    display_interface.window.blit(display_interface.s_proc, (5, 5))
```

Código da Versão 6.0

Arquivo: main.py

```
import data_and_funcs
import display_interface

import pygame
import time
import threading

def main():
    pygame.display.set_caption("PC Resources Monitor")
    pygame.display.init()

    ### Display ###
    clock = pygame.time.Clock()
    count = 0
    close = False
    files_page_menu = '1'
    processes_page_menu = '1'
    network_page_menu = '1'
    traffic_page_menu = '1'
    while not close:
        #t1 = time.perf_counter()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                close = True
            ### Keyboard ###
            if event.type == pygame.KEYDOWN:
                if ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'HOME')) or ((event.key ==
pygame.K_LEFT) and (display_interface.menu == 'FILES')):
                    display_interface.menu = 'CPU'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'CPU')) or ((event.key
== pygame.K_LEFT) and (display_interface.menu == 'PROCESS')):
                    display_interface.menu = 'FILES'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'FILES')) or ((event.key
== pygame.K_LEFT) and (display_interface.menu == 'NETWORK')):
                    display_interface.menu = 'PROCESS'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'PROCESS')) or
((event.key == pygame.K_LEFT) and (display_interface.menu == 'TRAFFIC')):
                    display_interface.menu = 'NETWORK'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'NETWORK')) or
((event.key == pygame.K_LEFT) and (display_interface.menu == 'HOME')):
                    display_interface.menu = 'TRAFFIC'
                elif ((event.key == pygame.K_RIGHT) and (display_interface.menu == 'TRAFFIC')) or
((event.key == pygame.K_LEFT) and (display_interface.menu == 'CPU')) or ((event.key == pygame.K_SPACE)):
                    display_interface.menu = 'HOME'
            ### Mouse ###
            if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                pos = pygame.mouse.get_pos()
                if display_interface.botton_display_0.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
                    display_interface.menu = 'HOME'
                elif display_interface.botton_display_1.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
                    display_interface.menu = 'CPU'
                elif display_interface.botton_display_2.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
                    display_interface.menu = 'FILES'
                elif display_interface.botton_display_3.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
```

```
                    display_interface.menu = 'PROCESS'
                elif display_interface.botton_display_4.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
                    display_interface.menu = 'NETWORK'
                elif display_interface.botton_display_5.collidepoint(((pos[0] -
display_interface.pos_x_s_display), (pos[1] - display_interface.pos_y_s_display))):
                    display_interface.menu = 'TRAFFIC'
        count += 1
        if count == 10:
            display_interface.menu_buttons(display_interface.menu)
            if display_interface.menu == 'HOME':
                #t1 = time.perf_counter()
                data_and_funcs.cpu_graph_0()
                data_and_funcs.mem_graph_0()
                data_and_funcs.hd_graph_0()
                data_and_funcs.pc_1()
                #t2 = time.perf_counter()
                #print("Process Time on Home Page: " + str(t2 - t1))

            elif display_interface.menu == 'CPU':
                #t1 = time.perf_counter()
                CPU_thrad_1 = threading.Thread(target=data_and_funcs.cpu_graph_0)
                CPU_thrad_2 = threading.Thread(target=data_and_funcs.cpu_graph_1)
                CPU_thrad_3 = threading.Thread(target=data_and_funcs.cpu_inf)
                CPU_thrad_1.start()
                CPU_thrad_2.start()
                CPU_thrad_3.start()
                CPU_thrad_1.join()
                CPU_thrad_2.join()
                CPU_thrad_3.join()
                #t2 = time.perf_counter()
                #print("Process Time on CPU Page: " + str(t2 - t1))

            elif display_interface.menu =='FILES':
                #t1 = time.perf_counter()
                list_files = threading.Thread(target=data_and_funcs.create_files_pages)
                FILES_thrad_1 = threading.Thread(target=data_and_funcs.files_display, args=files_page_menu)
                FILES_thrad_2 = threading.Thread(target=data_and_funcs.files_page_header,
args=files_page_menu)
                list_files.start()
                FILES_thrad_1.start()
                FILES_thrad_2.start()
                list_files.join()
                FILES_thrad_1.join()
                FILES_thrad_2.join()
                if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                    pos = pygame.mouse.get_pos()
                    #print(pos)
                    if data_and_funcs.files_next_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(files_page_menu) + 1) <= data_and_funcs.files_pages_count():
                            newpage = int(files_page_menu) + 1
                            files_page_menu = str(newpage)
                            #print("Displaying Files Page:", files_page_menu)
                    if data_and_funcs.files_back_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(files_page_menu) - 1) > 0:
                            newpage = int(files_page_menu) - 1
                            files_page_menu = str(newpage)
                            #print("Displaying Files Page:", files_page_menu)
                #t2 = time.perf_counter()
```

```
                #print("Process Time on FILES Page: " + str(t2 - t1))

            elif display_interface.menu == 'PROCESS':
                #t1 = time.perf_counter()
                PROCESS_thrad_1 = threading.Thread(target=data_and_funcs.processes_display,
args=processes_page_menu)
                PROCESS_thrad_2 = threading.Thread(target=data_and_funcs.processes_page_header,
args=processes_page_menu)
                PROCESS_thrad_1.start()
                PROCESS_thrad_2.start()
                PROCESS_thrad_1.join()
                PROCESS_thrad_2.join()
                if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                    pos = pygame.mouse.get_pos()
                    #print(pos)
                    if data_and_funcs.files_next_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(processes_page_menu) + 1) <= data_and_funcs.processes_pages_number:
                            newpage = int(processes_page_menu) + 1
                            processes_page_menu = str(newpage)
                            #print("Displaying Processes Page:", processes_page_menu)
                    if data_and_funcs.files_back_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(processes_page_menu) - 1) > 0:
                            newpage = int(processes_page_menu) - 1
                            processes_page_menu = str(newpage)
                            #print("Displaying Processes Page:", processes_page_menu)
                #t2 = time.perf_counter()
                #print("Process Time on PROCESS Page: " + str(t2 - t1))

            elif display_interface.menu == 'NETWORK':
                #t1 = time.perf_counter()
                NETWORK_thrad_1 = threading.Thread(target=data_and_funcs.network_display,
args=(network_page_menu, data_and_funcs.network_pages,))
                NETWORK_thrad_2 = threading.Thread(target=data_and_funcs.network_page_header,
args=(network_page_menu, data_and_funcs.network_pages_number,))
                NETWORK_thrad_1.start()
                NETWORK_thrad_2.start()
                NETWORK_thrad_1.join()
                NETWORK_thrad_2.join()
                if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                    pos = pygame.mouse.get_pos()
                    #print(pos)
                    if data_and_funcs.files_next_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(network_page_menu) + 1) <= data_and_funcs.network_pages_number:
                            newpage = int(network_page_menu) + 1
                            network_page_menu = str(newpage)
                            #print("Displaying NETWORK Page:", network_page_menu)
                    if data_and_funcs.files_back_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(network_page_menu) - 1) > 0:
                            newpage = int(network_page_menu) - 1
                            network_page_menu = str(newpage)
                            #print("Displaying NETWORK Page:", network_page_menu)
                #t2 = time.perf_counter()
                #print("Process Time on NETWORK Page: " + str(t2 - t1))

            elif display_interface.menu == 'TRAFFIC':
                #t1 = time.perf_counter()
                data_and_funcs.traffices_display(traffic_page_menu)
```

```
                data_and_funcs.traffic_page_header(traffic_page_menu)

                if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                    pos = pygame.mouse.get_pos()
                    #print(pos)
                    if data_and_funcs.files_next_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(traffic_page_menu) + 1) <= data_and_funcs.traffic_pages_count():
                            newpage = int(traffic_page_menu) + 1
                            traffic_page_menu = str(newpage)
                            #print("Displaying TRAFFIC Page:", traffic_page_menu)
                    if data_and_funcs.files_back_page_buttom.collidepoint(((pos[0] -
data_and_funcs.pos_x_s_header), (pos[1] - data_and_funcs.pos_y_s_header))):
                        if (int(traffic_page_menu) - 1) > 0:
                            newpage = int(traffic_page_menu) - 1
                            traffic_page_menu = str(newpage)
                            #print("Displaying TRAFFIC Page:", traffic_page_menu)
                #t2 = time.perf_counter()
                #print("Process Time on TRAFFIC Page: " + str(t2 - t1))

            count = 0
        pygame.display.update()
        clock.tick(60)
    pygame.display.quit()
if __name__ == "__main__":
    main()
```

Arquivo: display_interface.py

```
import pygame

### Windows Config. ###
WINDOW_WIDTH = 800
WINDOW_HEIGHT = 600
MENU_BOTTOM_BORDER_WIDTH = WINDOW_WIDTH
MENU_BOTTOM_BORDER_HEIGHT = 40
MENU_BOTTOM_WIDTH = (MENU_BOTTOM_BORDER_WIDTH - 10)
MENU_BOTTOM_HEIGHT = (MENU_BOTTOM_BORDER_HEIGHT - 10)

window = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT + MENU_BOTTOM_BORDER_HEIGHT))

### Surfaces ###
s_menu_bottom = pygame.surface.Surface((MENU_BOTTOM_WIDTH, MENU_BOTTOM_HEIGHT))
s_menu_bottom_border = pygame.surface.Surface((MENU_BOTTOM_BORDER_WIDTH, MENU_BOTTOM_BORDER_HEIGHT))
s_display = pygame.surface.Surface((int(MENU_BOTTOM_WIDTH / 2), MENU_BOTTOM_HEIGHT))
s = pygame.surface.Surface((WINDOW_WIDTH, int((WINDOW_HEIGHT)/4)))
s_dir = pygame.surface.Surface((WINDOW_WIDTH-10, WINDOW_HEIGHT-5))
s_dir_border = pygame.surface.Surface((WINDOW_WIDTH, WINDOW_HEIGHT))
s_header = pygame.surface.Surface((WINDOW_WIDTH-10, 65))
s_proc = pygame.surface.Surface((WINDOW_WIDTH-10, WINDOW_HEIGHT-5))
s_proc_border = pygame.surface.Surface((WINDOW_WIDTH, WINDOW_HEIGHT))
s_info = pygame.surface.Surface(((WINDOW_WIDTH - 10), int((WINDOW_HEIGHT)/4) - 5))
s_graph_1 = pygame.surface.Surface((WINDOW_WIDTH, int(((WINDOW_HEIGHT)/4)*2)))

### Colors ###
red = (150,0,0)
green = (255,0,0)
blue = (0,0,255)
light_blue = (0,200,255)
black = (0,0,0)
```

```
white = (255, 255, 255)
grey = (150, 150, 150)
grey1 = (80, 80, 80)
grey2 = (45, 45, 45)
grey2 = (55, 55, 55)
grey_blue1 = (156, 181, 201)
grey_blue2 = (0, 55, 130)
dark_blue = (0, 0, 50)
sand = (255, 230, 185)
sand2 = (225, 190, 145)
dark_sand = (165, 145, 110)
background = (45, 45, 45)

### Fonts ###
pygame.font.init()
def make_font(fonts, size):
    available = pygame.font.get_fonts()
    # get_fonts() returns a list of lowercase spaceless font names
    choices = map(lambda x:x.lower().replace(' ', ''), fonts)
    for choice in choices:
        if choice in available:
            return pygame.font.SysFont(choice, size)
    return pygame.font.Font(None, size)
_cached_fonts = {}
def get_font(font_preferences, size):
    global _cached_fonts
    key = str(font_preferences) + '|' + str(size)
    font = _cached_fonts.get(key, None)
    if font == None:
        font = make_font(font_preferences, size)
        _cached_fonts[key] = font
    return font

font_preferences = ["Consolas","Rockwell", "Calibri", "Arial", ]

font_1 = get_font(font_preferences, 30)
font_2 = get_font(font_preferences, 20)
font_3 = get_font(font_preferences, 16)
font_4 = get_font(font_preferences, 12)

### Menu ###
### Menu Settings ###
pos_x = 125
pos_y = (int(MENU_BOTTOM_BORDER_HEIGHT / 3) + 2)
button_size_pressed = 8
button_size_nonpressed = 10

pos_x_s_display = 150
pos_y_s_display = WINDOW_HEIGHT + 5

def menu_buttons(menu):
    s_menu_bottom_border.fill(black)
    s_menu_bottom.fill(background)
    s_display.fill(background)

    #botton HOME
    pygame.draw.circle(s_display, black, (pos_x,pos_y), (button_size_nonpressed + 3))
    botton_display_0 = pygame.draw.circle(s_display, dark_sand, (pos_x,pos_y), button_size_nonpressed)
    #botton CPU
    pygame.draw.circle(s_display, black, ((pos_x + 50),pos_y), (button_size_nonpressed + 3))
    botton_display_1 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 50),pos_y),
```

```
button_size_nonpressed)
    #botton FILES
    pygame.draw.circle(s_display, black, ((pos_x + 100),pos_y), (button_size_nonpressed + 3))
    botton_display_2 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 100),pos_y),
button_size_nonpressed)
    #botton PROCESS
    pygame.draw.circle(s_display, black, ((pos_x + 150),pos_y), (button_size_nonpressed + 3))
    botton_display_3 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 150),pos_y),
button_size_nonpressed)
    #botton NETWORK
    pygame.draw.circle(s_display, black, ((pos_x + 200),pos_y), (button_size_nonpressed + 3))
    botton_display_4 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 200),pos_y),
button_size_nonpressed)
    #botton TRAFFIC
    pygame.draw.circle(s_display, black, ((pos_x + 250),pos_y), (button_size_nonpressed + 3))
    botton_display_5 = pygame.draw.circle(s_display, dark_sand, ((pos_x + 250),pos_y),
button_size_nonpressed)

    if menu == 'HOME':
        pygame.draw.circle(s_display, black, (pos_x,pos_y), (button_size_pressed + 3))
        botton_display_0 = pygame.draw.circle(s_display, grey1, (pos_x,pos_y), button_size_pressed)
    elif menu == 'CPU':
        pygame.draw.circle(s_display, black, ((pos_x + 50),pos_y), (button_size_pressed + 3))
        botton_display_1 = pygame.draw.circle(s_display, grey1, ((pos_x + 50),pos_y), button_size_pressed)
    elif menu =='FILES':
        pygame.draw.circle(s_display, black, ((pos_x + 100),pos_y), (button_size_pressed + 3))
        botton_display_2 = pygame.draw.circle(s_display, grey1, ((pos_x + 100),pos_y), button_size_pressed)
    elif menu == 'PROCESS':
        pygame.draw.circle(s_display, black, ((pos_x + 150),pos_y), (button_size_pressed + 3))
        botton_display_3 = pygame.draw.circle(s_display, grey1, ((pos_x + 150),pos_y), button_size_pressed)
    elif menu == 'NETWORK':
        pygame.draw.circle(s_display, black, ((pos_x + 200),pos_y), (button_size_pressed + 3))
        botton_display_4 = pygame.draw.circle(s_display, grey1, ((pos_x + 200),pos_y), button_size_pressed)
    elif menu == 'TRAFFIC':
        pygame.draw.circle(s_display, black, ((pos_x + 250),pos_y), (button_size_pressed + 3))
        botton_display_5 = pygame.draw.circle(s_display, grey1, ((pos_x + 250),pos_y), button_size_pressed)

    pages = font_4.render("Pages:", 10, dark_sand)
    s_menu_bottom.blit(pages, (20, 10))
    ass = font_4.render("Dev.: Fábio R. P. Nunes", 10, dark_sand)
    s_menu_bottom.blit(ass, (610, 10))
    window.blit(s_menu_bottom_border, (0, WINDOW_HEIGHT))
    window.blit(s_menu_bottom, (5, (WINDOW_HEIGHT + 5)))
    window.blit(s_display, (pos_x_s_display, pos_y_s_display))

    return botton_display_0, botton_display_1, botton_display_2, botton_display_3, botton_display_4,
botton_display_5

menu = 'HOME'
botton_display_0 = menu_buttons(menu)[0]
botton_display_1 = menu_buttons(menu)[1]
botton_display_2 = menu_buttons(menu)[2]
botton_display_3 = menu_buttons(menu)[3]
botton_display_4 = menu_buttons(menu)[4]
botton_display_5 = menu_buttons(menu)[5]
```

Arquivo: data_and_funcs.py

```
import display_interface
```

```
import pygame
import psutil
import cpuinfo
import platform
import subprocess
import threading
import socket
import os
import datetime
import nmap

### Static Data ###
# PC Inf 1
pc_name = platform.node()
operational_system = platform.system()
osVersion = platform.version()
arc = platform.machine()
processor = platform.processor()
cores = psutil.cpu_count(logical=True)
pc_partitions = psutil.disk_partitions(all)
primary_storage = pc_partitions[0][0]
#local = os.path.dirname(os.path.realpath(__file__))
local = 'D:\\Coding\\Python\\Projeto de Bloco\\Tests'

# IPs
#hosts_mapping_ready = True
#network_mapping_ready = True
#IPs_mapping_ready = True
def get_ip_addresses(family):
    for interface, snics in psutil.net_if_addrs().items():
        for snic in snics:
            if snic.family == family:
                yield (interface, snic.address)

def ping(hostname):
    args = []

    if operational_system == "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 host(subnetwork):
    print("Mapping", subnetwork, "subnetwork...\r")
    hosts_list = []
    return_codes = dict()
    for i in range(1, 255):
        host_test = (subnetwork + '{0}'.format(i))
        return_codes[subnetwork + '{0}'.format(i)] = ping(host_test)
        if i %20 == 0:
            print(".", end = "")
        if return_codes[subnetwork + '{0}'.format(i)] == 0:
            hosts_list.append(subnetwork + '{0}'.format(i))
    print("\nMapping ready...")

    hosts_mapping_ready = True
```

```
    print("hosts_mapping_ready:", hosts_mapping_ready)
    return hosts_list

def hosts_data(hosts_list):
    nm = nmap.PortScanner()
    IPs = dict()
    for i in hosts_list:
        print("Scanning", i, "IP...")
        nm.scan(i)
        IPs[i] = nm[i]

    IPs_Scanned = True
    IPs_mapping_ready = True
    return IPs

ipv4s = list(get_ip_addresses(socket.AF_INET))
ipv6s = list(get_ip_addresses(socket.AF_INET6))
mac_address = list(get_ip_addresses(psutil.AF_LINK))
subnetwork = ".".join(ipv4s[0][1].split('.')[0:3]) + '.'

hosts_scanned = host(subnetwork)
#hosts_scanned = ['192.168.0.1', '192.168.0.17', '192.168.0.32', '192.168.0.50']

IPs_Info = hosts_data(hosts_scanned)
#IPs_Info = {'192.168.0.1': {'hostnames': [{'name': '', 'type': ''}], 'addresses': {'ipv4': '192.168.0.1',
'mac': '8C:44:4F:73:C0:D1'}, 'vendor': {'8C:44:4F:73:C0:D1': 'Humax'}, 'status': {'state': 'up', 'reason':
'arp-response'}, 'tcp': {22: {'state': 'filtered', 'reason': 'no-response', 'name': 'ssh', 'product': '',
'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 80: {'state': 'open', 'reason': 'syn-ack', 'name':
'http', 'product': 'micro_httpd', 'version': '', 'extrainfo': '', 'conf': '10', 'cpe':
'cpe:/a:acme:micro_httpd'}, 139: {'state': 'filtered', 'reason': 'no-response', 'name': 'netbios-ssn',
'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 5431: {'state': 'open', 'reason':
'syn-ack', 'name': 'upnp', 'product': 'Belkin/Linksys wireless router UPnP', 'version': '', 'extrainfo':
'UPnP 1.0; BRCM400 1.0', 'conf': '10', 'cpe': 'cpe:/o:linux:linux_kernel:2.4'}, 12345: {'state':
'filtered', 'reason': 'no-response', 'name': 'netbus', 'product': '', 'version': '', 'extrainfo': '',
'conf': '3', 'cpe': ''}}}, '192.168.0.17': {'hostnames': [{'name': '', 'type': ''}], 'addresses': {'ipv4':
'192.168.0.17', 'mac': '1A:32:21:02:17:E1'}, 'vendor': {}, 'status': {'state': 'up', 'reason': 'arp-
response'}}, '192.168.0.32': {'hostnames': [{'name': '', 'type': ''}], 'addresses': {'ipv4':
'192.168.0.32'}, 'vendor': {}, 'status': {'state': 'up', 'reason': 'localhost-response'}, 'tcp': {135:
{'state': 'open', 'reason': 'syn-ack', 'name': 'msrpc', 'product': 'Microsoft Windows RPC', 'version': '',
'extrainfo': '', 'conf': '10', 'cpe': 'cpe:/o:microsoft:windows'}, 139: {'state': 'open', 'reason': 'syn-
ack', 'name': 'netbios-ssn', 'product': 'Microsoft Windows netbios-ssn', 'version': '', 'extrainfo': '',
'conf': '10', 'cpe': 'cpe:/o:microsoft:windows'}, 445: {'state': 'open', 'reason': 'syn-ack', 'name':
'microsoft-ds', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}}}, '192.168.0.50':
{'hostnames': [{'name': '', 'type': ''}], 'addresses': {'ipv4': '192.168.0.50', 'mac':
'54:B8:0A:9C:DF:E8'}, 'vendor': {'54:B8:0A:9C:DF:E8': 'D-Link International'}, 'status': {'state': 'up',
'reason': 'arp-response'}, 'tcp': {23: {'state': 'filtered', 'reason': 'no-response', 'name': 'telnet',
'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 80: {'state': 'open', 'reason':
'syn-ack', 'name': 'http', 'product': 'mini_httpd', 'version': '1.19 19dec2003', 'extrainfo': '', 'conf':
'10', 'cpe': 'cpe:/a:acme:mini_httpd:1.19_19dec2003'}, 139: {'state': 'open', 'reason': 'syn-ack', 'name':
'tcpwrapped', 'product': '', 'version': '', 'extrainfo': '', 'conf': '8', 'cpe': ''}, 1050: {'state':
'open', 'reason': 'syn-ack', 'name': 'http', 'product': 'mini_httpd', 'version': '1.19 19dec2003',
'extrainfo': '', 'conf': '10', 'cpe': 'cpe:/a:acme:mini_httpd:1.19_19dec2003'}}}}
#IPs_Info = None

### Dynamic Data ###
def cpu_graph_0():
    ### DATA ###
    cpu = int(psutil.cpu_percent(interval=0.1))
    freq_per_cent = int(psutil.cpu_percent(interval=0.1))
    freq_max = psutil.cpu_freq().max
    freq_current = ((freq_max * freq_per_cent)/100)
```

```
    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*cpu) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,54), (790, 54), 6)

    bar_text = "CPU Usage"
    bar_text_2 = "- Current Frequency (Mhz): " + str(freq_current)
    bar_text_inside_01 = str(cpu) + "%"
    bar_text_inside_02 = str((100 - cpu)) + "%"


    text = display_interface.font_1.render(bar_text, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)

    display_interface.s.blit(text, (20, 18))
    display_interface.s.blit(text_2, (185, 26))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (700, 110))

    display_interface.window.blit(display_interface.s, (0, 0))

def cpu_graph_1():
    ### Data ###
    cores_percent = psutil.cpu_percent(interval=1, percpu=True)

    display_interface.s_graph_1.fill(display_interface.background)

    cores = len(cores_percent)

    x = y = shift = 10

    height = int(display_interface.s_graph_1.get_height() - 2*y)
    width = int((display_interface.s_graph_1.get_width() - 2*y - (cores + 1)*shift ) / cores)

    d = x + shift
    cores_count = 1
    for i in cores_percent:
        pygame.draw.rect(display_interface.s_graph_1, display_interface.black, (d+5, y+5, width, height))
        pygame.draw.rect(display_interface.s_graph_1, display_interface.red, (d, y, width, height))
        pygame.draw.rect(display_interface.s_graph_1, display_interface.grey_blue2, (d, y, width, ((1-
i/100)*height)))
        bar_text_inside_0 = str(int(i)) + "%"
        text_inside_0 = display_interface.font_3.render(bar_text_inside_0, 10, display_interface.sand2)
        display_interface.s_graph_1.blit(text_inside_0, ((d + (width/3)), (height - 10)))
        bar_text_inside_1 = "#"+str(cores_count)
        text_inside_1 = display_interface.font_3.render(bar_text_inside_1, 10, display_interface.dark_sand)
        display_interface.s_graph_1.blit(text_inside_1, ((d + (width/3)), y+10))
        cores_count += 1
        d = d + width + shift
```

```
    display_interface.window.blit(display_interface.s_graph_1, (0, (display_interface.WINDOW_HEIGHT)/4))

def mem_graph_0():
    ### DATA ###
    mem = psutil.virtual_memory()
    memTotalGigas = round((mem.total / (1024*1024*1024)), 2)
    memUsedGigas = round((mem.used / (1024*1024*1024)), 2)
    memFreeGigas = round((mem.free / (1024*1024*1024)), 2)
    mem_per_cent = int(mem.percent)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))

    width_bar2 = int((width_bar1*mem.percent) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH - 10), 45), 6)

    bar_text_1 = "Memory Usage"
    bar_text_2 = "(Total: " + str(memTotalGigas) + " GB)"
    bar_text_inside_01 = str(memUsedGigas) + " Used"
    bar_text_inside_02 = str(memFreeGigas) + " Free"
    bar_text_inside_03 = str(mem_per_cent) + "%"
    bar_text_inside_04 = str(100 - mem_per_cent) + "%"

    text_1 = display_interface.font_1.render(bar_text_1, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)
    text_inside3 = display_interface.font_2.render(bar_text_inside_03, 10, display_interface.sand2)
    text_inside4 = display_interface.font_2.render(bar_text_inside_04, 10, display_interface.dark_sand)

    display_interface.s.blit(text_1, (20, 10))
    display_interface.s.blit(text_2, (235, 18))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (655, 110))
    display_interface.s.blit(text_inside3, (35, 75))
    display_interface.s.blit(text_inside4, (730, 75))

    display_interface.window.blit(display_interface.s, (0, ((display_interface.WINDOW_HEIGHT)/4)))

def hd_graph_0():
    ### DATA ###
    hd = psutil.disk_usage(str(primary_storage))
    hdTotalGigas = round((hd.total / (1024*1024*1024)), 2)
    hdUsedGigas = round((hd.used / (1024*1024*1024)), 2)
    hdFreeGigas = round((hd.free / (1024*1024*1024)), 2)
    hd_per_cent = int(hd.percent)

    width_bar1 = int(display_interface.WINDOW_WIDTH - 2*20)
    display_interface.s.fill(display_interface.background)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar1, 70))
    pygame.draw.rect(display_interface.s, display_interface.grey_blue2, (20, 65, width_bar1, 70))
```

```
    width_bar2 = int((width_bar1*hd.percent) / 100)

    pygame.draw.rect(display_interface.s, display_interface.black, (25, 70, width_bar2, 70))
    pygame.draw.rect(display_interface.s, display_interface.red, (20, 65, width_bar2, 70))

    pygame.draw.line(display_interface.s, display_interface.black, (10,45), (790, 45), 6)

    bar_text_1 = "Primary Storage Usage"
    bar_text_2 = "(Total: " + str(hdTotalGigas) + " GB)"
    bar_text_inside_01 = str(hdUsedGigas) + " Used"
    bar_text_inside_02 = str(hdFreeGigas) + " Free"
    bar_text_inside_03 = str(hd_per_cent) + "%"
    bar_text_inside_04 = str(100 - hd_per_cent) + "%"

    text_1 = display_interface.font_1.render(bar_text_1, 10, display_interface.grey)
    text_2 = display_interface.font_3.render(bar_text_2, 10, display_interface.grey)
    text_inside1 = display_interface.font_2.render(bar_text_inside_01, 10, display_interface.sand2)
    text_inside2 = display_interface.font_2.render(bar_text_inside_02, 10, display_interface.dark_sand)
    text_inside3 = display_interface.font_2.render(bar_text_inside_03, 10, display_interface.sand2)
    text_inside4 = display_interface.font_2.render(bar_text_inside_04, 10, display_interface.dark_sand)

    display_interface.s.blit(text_1, (20, 10))
    display_interface.s.blit(text_2, (390, 18))
    display_interface.s.blit(text_inside1, (35, 110))
    display_interface.s.blit(text_inside2, (640, 110))
    display_interface.s.blit(text_inside3, (35, 75))
    display_interface.s.blit(text_inside4, (730, 75))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*2))))

def pc_1():
    display_interface.s_info.fill(display_interface.background)
    display_interface.s.fill(display_interface.black)

    pygame.draw.line(display_interface.s_info, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH -20), 45), 6)
    pos_y = 10
    text_0 = "PC Information"
    text_1 = "PC Name: " + str(pc_name)
    text_2 = "Operating System: " + str(operational_system) + " " + str(osVersion)
    text_3 = "Network Information"

    ipv4s_text_pos_y = 85
    text_4 = "- IPv4s"
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)
    display_interface.s_info.blit(font_text_4, (20, ipv4s_text_pos_y))
    ipv4s_text_pos_y += 15
    for i in range(len(ipv4s)):
        text = ipv4s[i][0] + ": " + ipv4s[i][1]
        font_text = display_interface.font_4.render(text, 10, display_interface.dark_sand)
        display_interface.s_info.blit(font_text, (20, ipv4s_text_pos_y))
        ipv4s_text_pos_y += 15

    text_4 = "MAC Address: " + str(mac_address[0][1])

    ipv6s_text_pos_y = 55
    text_5 = "- IPv6s"
    font_text_5 = display_interface.font_4.render(text_5, 10, display_interface.dark_sand)
    display_interface.s_info.blit(font_text_5, (400, ipv6s_text_pos_y))
    ipv6s_text_pos_y += 15
    for i in range(len(ipv6s)):
```

```
        text = ipv6s[i][0] + ": " + ipv6s[i][1]
        font_text = display_interface.font_4.render(text, 10, display_interface.dark_sand)
        display_interface.s_info.blit(font_text, (400, ipv6s_text_pos_y))
        ipv6s_text_pos_y += 15

    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    font_text_1 = display_interface.font_4.render(text_1, 10, display_interface.grey)
    font_text_2 = display_interface.font_4.render(text_2, 10, display_interface.grey)
    font_text_3 = display_interface.font_4.render(text_3, 10, display_interface.dark_sand)
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)

    display_interface.s_info.blit(font_text_0, (20, pos_y))
    display_interface.s_info.blit(font_text_1, (275, pos_y))
    pos_y += 15
    display_interface.s_info.blit(font_text_2, (275, pos_y))
    pos_y += 30
    display_interface.s_info.blit(font_text_3, (20, pos_y))
    display_interface.s_info.blit(font_text_4, (20, 70))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*3))))
    display_interface.window.blit(display_interface.s_info, (5,
(((((display_interface.WINDOW_HEIGHT)/4)*3)+5))))

def cpu_inf():
    inf = cpuinfo.get_cpu_info()
    cpu_name = inf['brand_raw']
    cpu_arch = inf['arch']
    cpu_bits = inf['bits']

    p_cores = psutil.cpu_count(logical=False)
    l_cores = psutil.cpu_count()

    display_interface.s_info.fill(display_interface.background)
    display_interface.s.fill(display_interface.black)

    pygame.draw.line(display_interface.s_info, display_interface.black, (10,45),
((display_interface.WINDOW_WIDTH -20), 45), 6)

    text_0 = "CPU Information"
    text_1 = "CPU Name: " + str(cpu_name)
    text_2 = "Architecture: " + str(cpu_arch) + " (" + str(arc) + ")"
    text_3 = "Word Length: " + str(cpu_bits) + " bits"
    text_4 = "Physical Cores: " + str(p_cores)
    text_5 = "Logical Cores: " + str(l_cores)

    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    font_text_1 = display_interface.font_4.render(text_1, 10, display_interface.dark_sand)
    font_text_2 = display_interface.font_4.render(text_2, 10, display_interface.dark_sand)
    font_text_3 = display_interface.font_4.render(text_3, 10, display_interface.dark_sand)
    font_text_4 = display_interface.font_4.render(text_4, 10, display_interface.dark_sand)
    font_text_5 = display_interface.font_4.render(text_5, 10, display_interface.dark_sand)

    display_interface.s_info.blit(font_text_0, (20, 10))
    display_interface.s_info.blit(font_text_1, (20, 55))
    display_interface.s_info.blit(font_text_2, (20, 70))
    display_interface.s_info.blit(font_text_3, (20, 85))
    display_interface.s_info.blit(font_text_4, (20, 100))
    display_interface.s_info.blit(font_text_5, (20, 115))

    display_interface.window.blit(display_interface.s, (0, ((((display_interface.WINDOW_HEIGHT)/4)*3))))
    display_interface.window.blit(display_interface.s_info, (5,
```

```
(((((display_interface.WINDOW_HEIGHT)/4)*3)+5))))

### Files & Processes Page
pos_x_s_header = 5
pos_y_s_header = 5

def display_next_page_buttom():
    text_next_page = "Next"
    #pygame.draw.circle(s_header, black, ((WINDOW_WIDTH-40),40), (10 + 3))
    files_next_page_buttom = pygame.draw.circle(display_interface.s_header, display_interface.background,
((display_interface.WINDOW_WIDTH-50),40), 20)
    pygame.draw.polygon(display_interface.s_header, display_interface.black,
((display_interface.WINDOW_WIDTH-35,45),(display_interface.WINDOW_WIDTH-
55,35),(display_interface.WINDOW_WIDTH-55,55)))
    pygame.draw.polygon(display_interface.s_header, display_interface.dark_sand,
((display_interface.WINDOW_WIDTH-40,40),(display_interface.WINDOW_WIDTH-
60,30),(display_interface.WINDOW_WIDTH-60,50)))
    font_text_next_page = display_interface.font_3.render(text_next_page, 10, display_interface.dark_sand)
    display_interface.s_header.blit(font_text_next_page, ((display_interface.WINDOW_WIDTH-70), 10))
    return files_next_page_buttom

def display_back_page_buttom():
    text_next_page = "Back"
    #pygame.draw.circle(s_header, black, ((WINDOW_WIDTH-100),40), (10 + 3))
    files_back_page_buttom = pygame.draw.circle(display_interface.s_header, display_interface.background,
((display_interface.WINDOW_WIDTH-100),40), 20)
    pygame.draw.polygon(display_interface.s_header, display_interface.black,
((display_interface.WINDOW_WIDTH-110,45),(display_interface.WINDOW_WIDTH-
90,35),(display_interface.WINDOW_WIDTH-90,55)))
    pygame.draw.polygon(display_interface.s_header, display_interface.dark_sand,
((display_interface.WINDOW_WIDTH-115,40),(display_interface.WINDOW_WIDTH-
95,30),(display_interface.WINDOW_WIDTH-95,50)))
    font_text_next_page = display_interface.font_3.render(text_next_page, 10, display_interface.dark_sand)
    display_interface.s_header.blit(font_text_next_page, ((display_interface.WINDOW_WIDTH-120), 10))
    return files_back_page_buttom
files_next_page_buttom = display_next_page_buttom()
files_back_page_buttom = display_back_page_buttom()

## Files Page
def files_page_header(files_page_menu):
    display_interface.s_header.fill(display_interface.background)

    if (((files_pages_count()) > (1)) and ((int(files_page_menu)) < (files_pages_count()))):
        display_next_page_buttom()
    if files_page_menu != '1':
        display_back_page_buttom()

    pos_x = 20
    pos_y = 10
    text_0 = "Local Files"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = str(local)
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_1, (pos_x, pos_y))
    pos_y += 28

    pygame.draw.line(display_interface.s_header, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)
```

```
    display_interface.window.blit(display_interface.s_header, (pos_x_s_header, pos_y_s_header))

def dir_files():
    lista = os.listdir(local)
    files = {}
    for i in lista:
        filepath = os.path.join(local, i)
        if os.path.isfile(filepath):
            files[i] = {}
            files[i]['Size'] = os.stat(filepath).st_size
            files[i]['Created'] = datetime.datetime.fromtimestamp(os.stat(filepath).st_ctime)
            files[i]['Mod'] = datetime.datetime.fromtimestamp(os.stat(filepath).st_mtime)
    return files

def create_files_pages():
    files = dir_files()
    files_count = 1
    pages_count = 1
    pages = {}
    pages[pages_count] = {}
    for i in list(files):
        pages[pages_count][i] = {}
        pages[pages_count][i]['Size'] = files[i]['Size']
        pages[pages_count][i]['Created'] = files[i]['Created']
        pages[pages_count][i]['Mod'] = files[i]['Mod']
        files_count += 1
        if files_count > 12:
            pages_count += 1
            pages[pages_count] = {}
            files_count = 1
    return pages

def files_pages_count():
    return len(create_files_pages())

def files_display(files_page_menu):
    display_interface.s_dir.fill(display_interface.background)
    display_interface.s_dir_border.fill(display_interface.black)

    files = create_files_pages()[int(files_page_menu)]

    pos_x_files = 47
    pos_y_files = 78
    files_count = 0
    for i in list(files):
        file_text = '{:^0.34}'.format(str(i))
        file_size_text = "Size: " + '{:^0.38}'.format(str(files[i]['Size']) + " bytes")
        file_created_text = "Created: " + '{:^0.38}'.format(str(files[i]['Created']))
        file_mod_text = "Modification: "+ '{:^0.38}'.format(str(files[i]['Mod']))

        font_file_text = display_interface.font_3.render(file_text, 10, display_interface.dark_sand)
        font_file_size_text = display_interface.font_4.render(file_size_text, 10,
display_interface.dark_sand)
        font_file_created_text = display_interface.font_4.render(file_created_text, 10,
display_interface.dark_sand)
        font_file_mod_text = display_interface.font_4.render(file_mod_text, 10,
display_interface.dark_sand)

        pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files, pos_y_files+3,
int((display_interface.WINDOW_WIDTH)/2)-75, (73)))
        pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files-5, pos_y_files-4,
```

```
int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
        pygame.draw.rect(display_interface.s_dir, display_interface.grey2, (pos_x_files-4, pos_y_files-3,
int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
        display_interface.s_dir.blit(font_file_text, (pos_x_files + 8, pos_y_files))
        pygame.draw.circle(display_interface.s_dir, display_interface.black, ((pos_x_files-5),(pos_y_files
+ 8)), 8)
        pygame.draw.circle(display_interface.s_dir, display_interface.grey_blue2, ((pos_x_files-
5),(pos_y_files + 8)), 6)
        pos_y_files += 18
        pygame.draw.rect(display_interface.s_dir, display_interface.black, (pos_x_files-5, pos_y_files,
int((display_interface.WINDOW_WIDTH)/2)-75, (53)))
        pygame.draw.rect(display_interface.s_dir, display_interface.grey2, (pos_x_files-4, pos_y_files+1,
int((display_interface.WINDOW_WIDTH)/2)-77, (51)))
        pos_y_files += 5
        display_interface.s_dir.blit(font_file_size_text, (pos_x_files, pos_y_files))
        pos_y_files += 15
        display_interface.s_dir.blit(font_file_created_text, (pos_x_files, pos_y_files))
        pos_y_files += 15
        display_interface.s_dir.blit(font_file_mod_text, (pos_x_files, pos_y_files))
        pos_y_files += 30

        files_count += 1
        if (files_count == 6):
            pos_x_files += int(((display_interface.WINDOW_WIDTH)/2)-22)
            pos_y_files = 78

    display_interface.window.blit(display_interface.s_dir_border, (0, 0))
    display_interface.window.blit(display_interface.s_dir, (5, 5))

### Processes Page
def processes_page_header(processes_page_menu):
    display_interface.s_header.fill(display_interface.background)

    if (((processes_pages_number) > (1)) and ((int(processes_page_menu)) < (processes_pages_number))):
        display_next_page_buttom()
    if processes_page_menu != '1':
        display_back_page_buttom()

    pos_x = 20
    pos_y = 10
    text_0 = "Process"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = "Running a total of: " + str(len(runned_process)) + " process"
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_1, (pos_x, pos_y))
    pos_y += 28

    pygame.draw.line(display_interface.s_header, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)

    display_interface.window.blit(display_interface.s_header, (pos_x_s_header, pos_y_s_header))

def running_processes():
    processes = psutil.pids()
    process = {}
    for i in processes:
        try:
            proc_name = str(psutil.Process(i).name())
            if proc_name not in process:
```

```
                process[proc_name] = {}
                process[proc_name]['PIDs'] = []
                process[proc_name]['Status'] = []
                process[proc_name]['# Total'] = 0
                process[proc_name]['Connections'] = []
            process[proc_name]['PIDs'].append(i)
            process[proc_name]['Status'].append(psutil.Process(i).status())
            process[proc_name]['# Total'] += 1
        except (psutil.AccessDenied, psutil.ZombieProcess, psutil.NoSuchProcess):
            continue
    return process
runned_process = running_processes()

def create_processes_pages():
    process = runned_process
    process_count = 1
    pages_count = 1
    pages = {}
    pages[pages_count] = {}
    for i in list(process):
        pages[pages_count][i] = {}
        pages[pages_count][i]['PIDs'] = process[i]['PIDs']
        pages[pages_count][i]['Status'] = process[i]['Status']
        pages[pages_count][i]['# Total'] = process[i]['# Total']
        process_count += 1
        if process_count > 12:
            pages_count += 1
            pages[pages_count] = {}
            process_count = 1
    return pages
processes_pages = create_processes_pages()

def processes_pages_count():
    return len(processes_pages)
processes_pages_number = processes_pages_count()

def processes_display(processes_page_menu):
    display_interface.s_proc.fill(display_interface.background)
    display_interface.s_proc_border.fill(display_interface.black)

    process = processes_pages[int(processes_page_menu)]

    pos_x_process = 47
    pos_y_process = 78
    process_count = 0
    for i in list(process):
        process_text = str(i)
        process_size_text = "PIDs: " + '{:^0.38}'.format(str(process[i]['PIDs']))
        process_created_text = "Status: " + '{:^0.37}'.format(str(process[i]['Status']))
        process_mod_text = "# Total: "+ '{:^0.38}'.format(str(process[i]['# Total']))

        font_process_text = display_interface.font_3.render(process_text, 10, display_interface.dark_sand)
        font_process_size_text = display_interface.font_4.render(process_size_text, 10,
display_interface.dark_sand)
        font_process_created_text = display_interface.font_4.render(process_created_text, 10,
display_interface.dark_sand)
        font_process_mod_text = display_interface.font_4.render(process_mod_text, 10,
display_interface.dark_sand)

        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process,
pos_y_process+3, int((display_interface.WINDOW_WIDTH)/2)-75, (73)))
```

```
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process-5,
pos_y_process-4, int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_process-4,
pos_y_process-3, int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
        display_interface.s_proc.blit(font_process_text, (pos_x_process + 8, pos_y_process))
        pygame.draw.circle(display_interface.s_proc, display_interface.black, ((pos_x_process-
5),(pos_y_process + 8)), 8)
        pygame.draw.circle(display_interface.s_proc, display_interface.grey_blue2, ((pos_x_process-
5),(pos_y_process + 8)), 6)
        pos_y_process += 18
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_process-5,
pos_y_process, int((display_interface.WINDOW_WIDTH)/2)-75, (53)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_process-4,
pos_y_process+1, int((display_interface.WINDOW_WIDTH)/2)-77, (51)))
        pos_y_process += 5
        display_interface.s_proc.blit(font_process_size_text, (pos_x_process, pos_y_process))
        pos_y_process += 15
        display_interface.s_proc.blit(font_process_created_text, (pos_x_process, pos_y_process))
        pos_y_process += 15
        display_interface.s_proc.blit(font_process_mod_text, (pos_x_process, pos_y_process))
        pos_y_process += 30

        process_count += 1
        if (process_count == 6):
            pos_x_process += int(((display_interface.WINDOW_WIDTH)/2)-22)
            pos_y_process = 78

        display_interface.window.blit(display_interface.s_proc_border, (0, 0))
        display_interface.window.blit(display_interface.s_proc, (5, 5))

### Network Page
def network_page_header(network_page_menu, network_pages_number):
    display_interface.s_header.fill(display_interface.background)

    if (((network_pages_number) > (1)) and ((int(network_page_menu)) < (network_pages_number))):
        display_next_page_buttom()
    if network_page_menu != '1':
        display_back_page_buttom()

    pos_x = 20
    pos_y = 10
    text_0 = "Network Information"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = "Total of: " + str(((len(hosts_scanned)) - 1)) + " Others Local IPs on " + subnetwork + "
Subnetwork"
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_1, (pos_x, pos_y))
    pos_y += 28

    pygame.draw.line(display_interface.s_header, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)

    display_interface.window.blit(display_interface.s_header, (pos_x_s_header, pos_y_s_header))

def hosts_info():
    IPs = dict()
    for i in hosts_scanned:
        IPs[i] = dict()
        try:
```

```
            IPs[i]['mac'] = IPs_Info[i]['addresses']['mac']
        except:
            pass
        try:
            IPs[i]['vendor'] = IPs_Info[i]['vendor'][IPs_Info[i]['addresses']['mac']]
        except:
            pass
        try:
            IPs[i]['response'] = IPs_Info[i]['status']['reason']
        except:
            pass
        try:
            IPs[i]['product'] = IPs_Info[i]['product']
        except:
            pass
    return IPs
IPs_data = hosts_info()

def create_network_pages(IPs_data):
    IPs = IPs_data
    IPs_count = 1
    pages_count = 1
    pages = dict()
    pages[pages_count] = dict()
    for i in list(IPs):
        if IPs[i]['response'] != 'localhost-response':
            pages[pages_count][i] = dict()
            try:
                pages[pages_count][i]['mac'] = IPs[i]['mac']
            except:
                pass
            try:
                pages[pages_count][i]['vendor'] = IPs[i]['vendor']
            except:
                pass
            try:
                pages[pages_count][i]['response'] = IPs[i]['response']
            except:
                pass

        IPs_count += 1
        if IPs_count > 12:
            pages_count += 1
            pages[pages_count] = {}
            IPs_count = 1
    return pages
network_pages = create_network_pages(IPs_data)

def network_pages_count(network_pages):
    return len(network_pages)
network_pages_number = network_pages_count(network_pages)

def network_display(network_page_menu, network_pages):
    display_interface.s_proc.fill(display_interface.background)
    display_interface.s_proc_border.fill(display_interface.black)

    pos_x_IPs = 47
    pos_y_IPs = 78
    IPs_count = 0

    IPs = network_pages[int(network_page_menu)]
```

```
    for i in list(IPs):
        IPs_text = str(i)
        try:
            IPs_size_text = "MAC: " + '{:^0.38}'.format(str(IPs[i]['mac']))
        except:
            pass
        try:
            IPs_created_text = "Vendor: " + '{:^0.37}'.format(str(IPs[i]['vendor']))
        except:
            pass
        try:
            IPs_mod_text = "Rensponse: "+ '{:^0.38}'.format(str(IPs[i]['response']))
        except:
            pass

        font_IPs_text = display_interface.font_3.render(IPs_text, 10, display_interface.dark_sand)
        font_IPs_size_text = display_interface.font_4.render(IPs_size_text, 10,
display_interface.dark_sand)
        font_IPs_created_text = display_interface.font_4.render(IPs_created_text, 10,
display_interface.dark_sand)
        font_IPs_mod_text = display_interface.font_4.render(IPs_mod_text, 10, display_interface.dark_sand)

        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_IPs, pos_y_IPs+3,
int((display_interface.WINDOW_WIDTH)/2)-75, (73)))
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_IPs-5, pos_y_IPs-4,
int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_IPs-4, pos_y_IPs-3,
int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
        display_interface.s_proc.blit(font_IPs_text, (pos_x_IPs + 8, pos_y_IPs))
        pygame.draw.circle(display_interface.s_proc, display_interface.black, ((pos_x_IPs-5),(pos_y_IPs +
8)), 8)
        pygame.draw.circle(display_interface.s_proc, display_interface.grey_blue2, ((pos_x_IPs-
5),(pos_y_IPs + 8)), 6)
        pos_y_IPs += 18
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_IPs-5, pos_y_IPs,
int((display_interface.WINDOW_WIDTH)/2)-75, (53)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_IPs-4, pos_y_IPs+1,
int((display_interface.WINDOW_WIDTH)/2)-77, (51)))
        pos_y_IPs += 5
        display_interface.s_proc.blit(font_IPs_size_text, (pos_x_IPs, pos_y_IPs))
        pos_y_IPs += 15
        display_interface.s_proc.blit(font_IPs_created_text, (pos_x_IPs, pos_y_IPs))
        pos_y_IPs += 15
        display_interface.s_proc.blit(font_IPs_mod_text, (pos_x_IPs, pos_y_IPs))
        pos_y_IPs += 30

        IPs_count += 1
        if (IPs_count == 6):
            pos_x_IPs += int(((display_interface.WINDOW_WIDTH)/2)-22)
            pos_y_IPs = 78

    display_interface.window.blit(display_interface.s_proc_border, (0, 0))
    display_interface.window.blit(display_interface.s_proc, (5, 5))

### TRAFFIC
def traffic_page_header(traffic_page_menu):
    display_interface.s_header.fill(display_interface.background)

    if (((traffic_pages_count()) > (1)) and ((int(traffic_page_menu)) < (traffic_pages_count()))):
        display_next_page_buttom()
```

```
    if traffic_page_menu != '1':
        display_back_page_buttom()

    pos_x = 20
    pos_y = 10
    text_0 = "Traffic Information"
    font_text_0 = display_interface.font_1.render(text_0, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_0, (pos_x, pos_y))
    pos_y += 30
    text_1 = "Traffic of " + str(len(traffic_interface_scanner())) + " Interfaces."
    font_text_1 = display_interface.font_3.render(text_1, 10, display_interface.grey)
    display_interface.s_header.blit(font_text_1, (pos_x, pos_y))
    pos_y += 28

    pygame.draw.line(display_interface.s_header, display_interface.black, (10,60),
((display_interface.WINDOW_WIDTH -20), 60), 6)

    display_interface.window.blit(display_interface.s_header, (pos_x_s_header, pos_y_s_header))

def traffic_interface_scanner():
    io_status = psutil.net_io_counters(pernic=True)
    interface_names = list()
    for i in io_status:
        interface_names.append(str(i))

    #print(interface_names)

    interfaces_traffic = dict()
    for j in interface_names:
        interfaces_traffic[j] = dict()
        interfaces_traffic[j]['bytes_sent'] = io_status[j][0]
        interfaces_traffic[j]['bytes_recv'] = io_status[j][1]
        interfaces_traffic[j]['packets_sent'] = io_status[j][2]
        interfaces_traffic[j]['packets_recv'] = io_status[j][3]

    return interfaces_traffic

def traffic_process_scanner():
    processes = psutil.pids()

    process = {}
    for i in processes:
        try:
            proc_name = str(psutil.Process(i).name())
            if proc_name not in process:
                process[proc_name] = {}
                process[proc_name]['Connections'] = []
            if psutil.Process(i).connections() != []:
                process[proc_name]['Connections'].append(psutil.Process(i).connections())

        except (psutil.AccessDenied, psutil.ZombieProcess, psutil.NoSuchProcess):
            continue

    process_traffic = dict()
    for i in process:
        try:
            if process[i]['Connections'] == []:
                process[i]['Connections'] = 'No Contection'

            if process[i]['Connections'] != 'No Contection':
                process_traffic[i] = dict()
```

```
                process_traffic[i]['Conection Status'] = process[i]['Connections'][0][1][5]
                process_traffic[i]['Local Addr'] = process[i]['Connections'][0][1][3][0]
                process_traffic[i]['Local Port'] = process[i]['Connections'][0][1][3][1]

                if process[i]['Connections'][0][1][4] != ():
                    process_traffic[i]['Remote Addr'] = process[i]['Connections'][0][1][4][0]
                    process_traffic[i]['Remote Port'] = process[i]['Connections'][0][1][4][1]
                elif process[i]['Connections'][0][1][4] == ():
                    process_traffic[i]['Remote Addr'] = "None"
                    process_traffic[i]['Remote Port'] = "None"
        except:
            continue

    return process_traffic
traffic_processes = traffic_process_scanner()

def create_traffic_interface_pages():
    traffic_data = traffic_interface_scanner()
    traffic_data_count = 1
    pages_count = 1
    pages = dict()
    pages[pages_count] = dict()
    for i in list(traffic_data):
        pages[pages_count][i] = dict()
        pages[pages_count][i]['bytes_sent'] = traffic_data[i]['bytes_sent']
        pages[pages_count][i]['bytes_recv'] = traffic_data[i]['bytes_recv']
        pages[pages_count][i]['packets_sent'] = traffic_data[i]['packets_sent']
        pages[pages_count][i]['packets_recv'] = traffic_data[i]['packets_recv']
        traffic_data_count += 1
        if traffic_data_count > 12:
            pages_count += 1
            pages[pages_count] = {}
            traffic_data_count = 1
    return pages

def create_traffic_process_pages():
    traffic_data = traffic_processes
    traffic_data_count = 1
    pages_count = 1
    pages = dict()
    pages[pages_count] = dict()
    for i in list(traffic_data):
        pages[pages_count][i] = dict()
        try:
            pages[pages_count][i]['Conection Status'] = traffic_data[i]['Conection Status']
        except:
            continue
        try:
            pages[pages_count][i]['Local Addr'] = traffic_data[i]['Local Addr']
        except:
            continue
        try:
            pages[pages_count][i]['Local Port'] = traffic_data[i]['Local Port']
        except:
            continue
        try:
            pages[pages_count][i]['Remote Addr'] = traffic_data[i]['Remote Addr']
        except:
            continue
        try:
            pages[pages_count][i]['Remote Port'] = traffic_data[i]['Remote Port']
```

```
        except:
            continue
        traffic_data_count += 1
        if traffic_data_count > 12:
            pages_count += 1
            pages[pages_count] = {}
            traffic_data_count = 1
    return pages

def traffic_interface_pages_count():
    return len(create_traffic_interface_pages())
def traffic_process_pages_count():
    return len(create_traffic_process_pages())
def traffic_pages_count():
    return traffic_interface_pages_count() + traffic_process_pages_count()

def traffices_display(traffic_page_menu):
    display_interface.s_proc.fill(display_interface.background)
    display_interface.s_proc_border.fill(display_interface.black)

    traffic = create_traffic_interface_pages()[int(traffic_page_menu)]

    pos_x_traffic = 47
    pos_y_traffic = 78
    traffic_count = 0
    for i in list(traffic):
        traffic_text = str(i)
        traffic_sent_text = "Bytes Sent (Bytes): " + '{:^0.38}'.format(str(traffic[i]['bytes_sent']))
        traffic_recv_text = "Bytes Received (Bytes): " + '{:^0.37}'.format(str(traffic[i]['bytes_recv']))
        traffic_pck_sent_text = "Packets Sent: "+ '{:^0.38}'.format(str(traffic[i]['packets_sent']))
        traffic_pck_recv_text = "Packets Received: "+ '{:^0.38}'.format(str(traffic[i]['packets_sent']))

        font_traffic_text = display_interface.font_3.render(traffic_text, 10, display_interface.dark_sand)
        font_traffic_sent_text = display_interface.font_4.render(traffic_sent_text, 10,
display_interface.dark_sand)
        font_traffic_recv_text = display_interface.font_4.render(traffic_recv_text, 10,
display_interface.dark_sand)
        font_traffic_pck_sent_text = display_interface.font_4.render(traffic_pck_sent_text, 10,
display_interface.dark_sand)
        font_traffic_pck_recv_text = display_interface.font_4.render(traffic_pck_recv_text, 10,
display_interface.dark_sand)

        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_traffic,
pos_y_traffic+3, int((display_interface.WINDOW_WIDTH)/2)-75, (86)))
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_traffic-5,
pos_y_traffic-4, int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_traffic-4,
pos_y_traffic-3, int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
        display_interface.s_proc.blit(font_traffic_text, (pos_x_traffic + 8, pos_y_traffic))
        pygame.draw.circle(display_interface.s_proc, display_interface.black, ((pos_x_traffic-
5),(pos_y_traffic + 8)), 8)
        pygame.draw.circle(display_interface.s_proc, display_interface.grey_blue2, ((pos_x_traffic-
5),(pos_y_traffic + 8)), 6)
        pos_y_traffic += 18
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_traffic-5,
pos_y_traffic, int((display_interface.WINDOW_WIDTH)/2)-75, (66)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_traffic-4,
pos_y_traffic+1, int((display_interface.WINDOW_WIDTH)/2)-77, (64)))
        pos_y_traffic += 5
        display_interface.s_proc.blit(font_traffic_sent_text, (pos_x_traffic, pos_y_traffic))
        pos_y_traffic += 15
```

```
        display_interface.s_proc.blit(font_traffic_recv_text, (pos_x_traffic, pos_y_traffic))
        pos_y_traffic += 15
        display_interface.s_proc.blit(font_traffic_pck_sent_text, (pos_x_traffic, pos_y_traffic))
        pos_y_traffic += 15
        display_interface.s_proc.blit(font_traffic_pck_recv_text, (pos_x_traffic, pos_y_traffic))
        pos_y_traffic += 30

        traffic_count += 1
        if (traffic_count == 6):
            pos_x_traffic += int(((display_interface.WINDOW_WIDTH)/2)-22)
            pos_y_traffic = 78

    traffic = create_traffic_process_pages()[int(traffic_page_menu)]

    for i in list(traffic):
        traffic_text = str(i)
        try:
            traffic_conection_text = "Conection Status: " + '{:^0.38}'.format(str(traffic[i]['Conection
Status']))
        except:
            continue
        try:
            traffic_laddr_text = "Local Addr: " + '{:^0.37}'.format(str(traffic[i]['Local Addr']))
        except:
            continue
        try:
            traffic_lport_sent_text = "Local Port "+ '{:^0.38}'.format(str(traffic[i]['Local Port']))
        except:
            continue
        try:
            traffic_raddr_recv_text = "Remote Addr: "+ '{:^0.38}'.format(str(traffic[i]['Remote Addr']))
        except:
            continue
        try:
            traffic_rport_recv_text = "Remote Port: "+ '{:^0.38}'.format(str(traffic[i]['Remote Port']))
        except:
            continue
        font_traffic_text = display_interface.font_3.render(traffic_text, 10, display_interface.dark_sand)
        font_traffic_conection_text = display_interface.font_4.render(traffic_conection_text, 10,
display_interface.dark_sand)
        font_traffic_laddr_text = display_interface.font_4.render(traffic_laddr_text, 10,
display_interface.dark_sand)
        font_traffic_lport_sent_text = display_interface.font_4.render(traffic_lport_sent_text, 10,
display_interface.dark_sand)
        font_traffic_raddr_recv_text = display_interface.font_4.render(traffic_raddr_recv_text, 10,
display_interface.dark_sand)
        font_traffic_rport_recv_text = display_interface.font_4.render(traffic_rport_recv_text, 10,
display_interface.dark_sand)

        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_traffic,
pos_y_traffic+3, int((display_interface.WINDOW_WIDTH)/2)-75, (96)))
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_traffic-5,
pos_y_traffic-4, int((display_interface.WINDOW_WIDTH)/2)-75, (22)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_traffic-4,
pos_y_traffic-3, int((display_interface.WINDOW_WIDTH)/2)-77, (20)))
        display_interface.s_proc.blit(font_traffic_text, (pos_x_traffic + 8, pos_y_traffic))
        pygame.draw.circle(display_interface.s_proc, display_interface.black, ((pos_x_traffic-
5),(pos_y_traffic + 8)), 8)
        pygame.draw.circle(display_interface.s_proc, display_interface.grey_blue2, ((pos_x_traffic-
5),(pos_y_traffic + 8)), 6)
        pos_y_traffic += 18
```

```
        pygame.draw.rect(display_interface.s_proc, display_interface.black, (pos_x_traffic-5,
pos_y_traffic, int((display_interface.WINDOW_WIDTH)/2)-75, (76)))
        pygame.draw.rect(display_interface.s_proc, display_interface.grey2, (pos_x_traffic-4,
pos_y_traffic+1, int((display_interface.WINDOW_WIDTH)/2)-77, (74)))
        pos_y_traffic += 5
        display_interface.s_proc.blit(font_traffic_conection_text, (pos_x_traffic, pos_y_traffic))
        pos_y_traffic += 15
        display_interface.s_proc.blit(font_traffic_laddr_text, (pos_x_traffic, pos_y_traffic))
        pos_y_traffic += 15
        display_interface.s_proc.blit(font_traffic_lport_sent_text, (pos_x_traffic, pos_y_traffic))
        pos_y_traffic += 15
        display_interface.s_proc.blit(font_traffic_raddr_recv_text, (pos_x_traffic, pos_y_traffic))
        pos_y_traffic += 15
        display_interface.s_proc.blit(font_traffic_rport_recv_text, (pos_x_traffic, pos_y_traffic))
        pos_y_traffic += 30

        traffic_count += 1
        if (traffic_count == 6):
            pos_x_traffic += int(((display_interface.WINDOW_WIDTH)/2)-22)
            pos_y_traffic = 78

        display_interface.window.blit(display_interface.s_proc_border, (0, 0))
        display_interface.window.blit(display_interface.s_proc, (5, 5))
```

Código da Versão 7.0

Arquivo: main.py

```
import functions, interface, database

import pygame
import time
import threading
import socket
import pickle
import os # Only used for os._exit(0)

def main():
    pygame.display.set_caption("PC Resources Monitor")
    pygame.display.init()
    ### Display ###
    clock = pygame.time.Clock()
    count = 0
    close = False
    files_page_menu_offline = '1'
    files_page_menu_online = '1'
    processes_page_menu = '1'
    processes_page_scroll_y = 5
    hosts_mapped = False
    first_connection = True
    connection_status = ' '
    data_ready = False
    while not close:
        if not hosts_mapped:
            thread_mapping = threading.Thread(target=database.subnetwork_mapping,
args=[database.operational_system_data, database.subnetwork])
            thread_mapping.start()
            hosts_mapped = True
        for event in pygame.event.get():
            # Quit
            if event.type == pygame.QUIT:
                close = True
                os._exit(0)
            # Keyboard Inputs
            if event.type == pygame.KEYDOWN:
                if ((event.key == pygame.K_RIGHT) and (interface.menu == 'HOME')) or ((event.key ==
pygame.K_LEFT) and (interface.menu == 'FILES')):
                    interface.menu = 'CPU'
                elif ((event.key == pygame.K_RIGHT) and (interface.menu == 'CPU')) or ((event.key ==
pygame.K_LEFT) and (interface.menu == 'PROCESS')):
                    interface.menu = 'FILES'
                elif ((event.key == pygame.K_RIGHT) and (interface.menu == 'FILES')) or ((event.key ==
pygame.K_LEFT) and (interface.menu == 'HOME')):
                    interface.menu = 'PROCESS'
                elif ((event.key == pygame.K_RIGHT) and (interface.menu == 'PROCESS')) or ((event.key ==
pygame.K_LEFT) and (interface.menu == 'CPU')) or ((event.key == pygame.K_SPACE)):
                    interface.menu = 'HOME'
                if ((event.key == pygame.K_TAB) and (interface.connection == 'OFFLINE')):
                    interface.connection = 'ONLINE'
                elif ((event.key == pygame.K_TAB) and (interface.connection == 'ONLINE')):
                    interface.connection = 'OFFLINE'
            # Mouse Inputs
            if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                pos = pygame.mouse.get_pos()
                if interface.botton_display_0.collidepoint(((pos[0] - interface.pos_x_s_display), (pos[1] -
interface.pos_y_s_display))):
```

```
                    interface.menu = 'HOME'
                elif interface.botton_display_1.collidepoint(((pos[0] - interface.pos_x_s_display), (pos[1]
- interface.pos_y_s_display))):
                    interface.menu = 'CPU'
                elif interface.botton_display_2.collidepoint(((pos[0] - interface.pos_x_s_display), (pos[1]
- interface.pos_y_s_display))):
                    interface.menu = 'FILES'
                elif interface.botton_display_3.collidepoint(((pos[0] - interface.pos_x_s_display), (pos[1]
- interface.pos_y_s_display))):
                    interface.menu = 'PROCESS'

                if interface.botton_online.collidepoint(((pos[0] - 5), (pos[1] -
interface.pos_y_s_display))):
                    interface.connection = 'ONLINE'

                elif interface.botton_offline.collidepoint(((pos[0] - 5), (pos[1] -
interface.pos_y_s_display))):
                    interface.connection = 'OFFLINE'
        # Pages Display
        count += 1
        if count == 5:
            if interface.connection == 'OFFLINE':
                if interface.menu == 'HOME':
                    functions.cpu_graph_1(interface.connection, database.cpu_percent_data,
database.cpu_max_freq_data)
                    functions.mem_graph(interface.connection, database.mem_data)
                    functions.hd_graph(interface.connection, database.hd_data)
                    functions.pc_cpu_inf(database.pc_name_data, database.operational_system_data,
database.os_version_data, database.cpu_data, database.arc_data, database.p_cores_data,
database.l_cores_data)
                elif interface.menu == 'CPU':
                    CPU_thrad_1 = threading.Thread(target=functions.cpu_graph_1,
args=[interface.connection, database.cpu_percent_data, database.cpu_max_freq_data])
                    CPU_thrad_2 = threading.Thread(target=functions.cpu_graph_2,
args=[interface.connection, database.cpu_cores_percent_data])
                    functions.pc_cpu_inf(database.pc_name_data, database.operational_system_data,
database.os_version_data, database.cpu_data, database.arc_data, database.p_cores_data,
database.l_cores_data)
                    CPU_thrad_1.start()
                    CPU_thrad_2.start()
                    CPU_thrad_1.join()
                    CPU_thrad_2.join()
                elif interface.menu =='FILES':
                    files_data = functions.scan_files(database.local_data)
                    files_pages_data = functions.create_files_pages(files_data)
                    files_pages_count_data = functions.files_pages_count(files_pages_data)
                    functions.files_display(files_pages_data, files_page_menu_offline)
                    functions.files_page_header(database.local_data, files_pages_count_data,
files_page_menu_offline)
                    if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                        pos = pygame.mouse.get_pos()
                        if interface.next_page_buttom.collidepoint(((pos[0] - interface.pos_x_s_header),
(pos[1] - interface.pos_y_s_header))):
                            if (int(files_page_menu_offline) + 1) <=
functions.files_pages_count(files_pages_data):
                                newpage = int(files_page_menu_offline) + 1
                                files_page_menu_offline = str(newpage)
                        if interface.back_page_buttom.collidepoint(((pos[0] - interface.pos_x_s_header),
(pos[1] - interface.pos_y_s_header))):
                            if (int(files_page_menu_offline) - 1) > 0:
                                newpage = int(files_page_menu_offline) - 1
```

```
                                files_page_menu_offline = str(newpage)
                elif interface.menu == 'PROCESS':
                    end_pos_y = functions.network_and_processes_display(interface.connection,
database.hosts_data, database.ips_data, database.processes_data, database.traffic_interface_data,
processes_page_scroll_y, processes_page_menu)
                    functions.network_and_processes_page_header(database.ipv4s_data,
database.mac_address_data, processes_page_menu)
                    if event.type == pygame.MOUSEBUTTONUP:
                        if  event.button == 4:
                            processes_page_scroll_y = min(processes_page_scroll_y + 10, 0)
                        if  event.button == 5:
                            processes_page_scroll_y = max(processes_page_scroll_y - 10, -(end_pos_y - 600))
                    if event.type == pygame.KEYDOWN:
                        if event.key == pygame.K_PAGEUP:
                            processes_page_scroll_y = min(processes_page_scroll_y + 60, 0)
                        elif event.key == pygame.K_PAGEDOWN:
                            processes_page_scroll_y = max(processes_page_scroll_y - 60, -(end_pos_y - 600))
                        elif event.key == pygame.K_HOME:
                            processes_page_scroll_y = 0
                        elif event.key == pygame.K_END:
                            processes_page_scroll_y = -(end_pos_y - 600)
                    if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                        pos = pygame.mouse.get_pos()
                        if interface.pg_end_buttom.collidepoint(((pos[0] - interface.pos_x_s_header),
(pos[1] - interface.pos_y_s_header))):
                            processes_page_scroll_y = -(end_pos_y - 600)
                        elif interface.pg_top_buttom.collidepoint(((pos[0] - interface.pos_x_s_header),
(pos[1] - interface.pos_y_s_header))):
                            processes_page_scroll_y = 0
                        elif interface.pg_dowm_buttom.collidepoint(((pos[0] - interface.pos_x_s_header),
(pos[1] - interface.pos_y_s_header))):
                            processes_page_scroll_y = max(processes_page_scroll_y - 60, -(end_pos_y - 600))
                        elif interface.pg_up_buttom.collidepoint(((pos[0] - interface.pos_x_s_header),
(pos[1] - interface.pos_y_s_header))):
                            processes_page_scroll_y = min(processes_page_scroll_y + 60, 0)

            elif interface.connection == 'ONLINE':
                if first_connection:
                    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                    try:
                        s.connect((socket.gethostname(), 666))
                        print("Connected.")
                        connection_status = 'OK'
                        first_connection = False
                    except Exception as error:
                        print(str(error))
                        connection_status = 'ERROR'
                    interface.connection = functions.connection_check(connection_status)
                else:
                    data_ready = False
                    try:
                        s.send(interface.menu.encode('utf-8'))
                        bytes = s.recv(51200)
                        data = pickle.loads(bytes)
                        data_ready = True
                        connection_status = 'OK'
                    except Exception as error:
                        print(str(error))
                        connection_status = 'ERROR'
                    interface.connection = functions.connection_check(connection_status)
                if data_ready:
```

```
                    if interface.menu == 'HOME':
                        functions.cpu_graph_1(interface.connection, data['cpu_percent'],
data['cpu_max_freq'])
                        functions.mem_graph(interface.connection, data['mem'])
                        functions.hd_graph(interface.connection, data['hd'])
                        functions.pc_cpu_inf(data['pc_name'], data['os'], data['os_version'],
data['cpu_info'], data['cpu_arc'], data['cpu_pcores'], data['cpu_lcores'])
                    elif interface.menu == 'CPU':
                        functions.cpu_graph_1(interface.connection, data['cpu_percent'],
data['cpu_max_freq'])
                        functions.cpu_graph_2(interface.connection, data['cpu_cores_percent'])
                        functions.pc_cpu_inf(data['pc_name'], data['os'], data['os_version'],
data['cpu_info'], data['cpu_arc'], data['cpu_pcores'], data['cpu_lcores'])
                    elif interface.menu =='FILES':
                        files_data = data['files']
                        files_pages_data = functions.create_files_pages(files_data)
                        files_pages_count_data = functions.files_pages_count(files_pages_data)
                        functions.files_display(files_pages_data, files_page_menu_online)
                        functions.files_page_header(data['local'], files_pages_count_data,
files_page_menu_online)
                        if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                            pos = pygame.mouse.get_pos()
                            if interface.next_page_buttom.collidepoint(((pos[0] -
interface.pos_x_s_header), (pos[1] - interface.pos_y_s_header))):
                                if (int(files_page_menu_online) + 1) <=
functions.files_pages_count(files_pages_data):
                                    newpage = int(files_page_menu_online) + 1
                                    files_page_menu_online = str(newpage)
                            if interface.back_page_buttom.collidepoint(((pos[0] -
interface.pos_x_s_header), (pos[1] - interface.pos_y_s_header))):
                                if (int(files_page_menu_online) - 1) > 0:
                                    newpage = int(files_page_menu_online) - 1
                                    files_page_menu_online = str(newpage)
                    elif interface.menu == 'PROCESS':
                        end_pos_y = functions.network_and_processes_display(interface.connection,
data['hosts'], data['ips'], data['processes'], data['traffic_interface'], processes_page_scroll_y,
processes_page_menu)
                        functions.network_and_processes_page_header(data['ipv4s'], data['mac_address'],
processes_page_menu)
                        if event.type == pygame.MOUSEBUTTONUP:
                            if  event.button == 4:
                                processes_page_scroll_y = min(processes_page_scroll_y + 10, 0)
                            if  event.button == 5:
                                processes_page_scroll_y = max(processes_page_scroll_y - 10, -(end_pos_y -
600))
                        if event.type == pygame.KEYDOWN:
                            if event.key == pygame.K_PAGEUP:
                                processes_page_scroll_y = min(processes_page_scroll_y + 60, 0)
                            elif event.key == pygame.K_PAGEDOWN:
                                processes_page_scroll_y = max(processes_page_scroll_y - 60, -(end_pos_y -
600))
                            elif event.key == pygame.K_HOME:
                                processes_page_scroll_y = 0
                            elif event.key == pygame.K_END:
                                processes_page_scroll_y = -(end_pos_y - 600)
                        if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                            pos = pygame.mouse.get_pos()
                            if interface.pg_end_buttom.collidepoint(((pos[0] - interface.pos_x_s_header),
(pos[1] - interface.pos_y_s_header))):
                                processes_page_scroll_y = -(end_pos_y - 600)
                            elif interface.pg_top_buttom.collidepoint(((pos[0] - interface.pos_x_s_header),
```

```
(pos[1] - interface.pos_y_s_header))):
                                processes_page_scroll_y = 0
                            elif interface.pg_dowm_buttom.collidepoint(((pos[0] -
interface.pos_x_s_header), (pos[1] - interface.pos_y_s_header))):
                                processes_page_scroll_y = max(processes_page_scroll_y - 60, -(end_pos_y -
600))
                            elif interface.pg_up_buttom.collidepoint(((pos[0] - interface.pos_x_s_header),
(pos[1] - interface.pos_y_s_header))):
                                processes_page_scroll_y = min(processes_page_scroll_y + 60, 0)
            interface.menu_buttons(interface.menu, interface.connection)
            count = 0
        pygame.display.update()
        clock.tick(60)
    pygame.display.quit()
if __name__ == "__main__":
    main()
```

Arquivo: interface.py

```
import pygame

### Functions ###
def menu_buttons(menu, connection):
    s_menu_bottom_border.fill(black)
    s_menu_bottom.fill(background)
    s_display.fill(background)

    pos_zero_x = 125
    pos_x = pos_zero_x
    pos_y = ((int(MENU_BOTTOM_BORDER_HEIGHT / 3) + 2) + 13)
    button_size_pressed = 8
    button_size_nonpressed = 10

    #botton HOME
    pygame.draw.circle(s_display, black, (pos_x, pos_y), (button_size_nonpressed + 3))
    botton_display_0 = pygame.draw.circle(s_display, dark_sand, (pos_x, pos_y), button_size_nonpressed)
    #botton CPU
    pos_x += 50
    pygame.draw.circle(s_display, black, (pos_x, pos_y), (button_size_nonpressed + 3))
    botton_display_1 = pygame.draw.circle(s_display, dark_sand, (pos_x, pos_y), button_size_nonpressed)
    #botton FILES
    pos_x += 50
    pygame.draw.circle(s_display, black, (pos_x, pos_y), (button_size_nonpressed + 3))
    botton_display_2 = pygame.draw.circle(s_display, dark_sand, (pos_x, pos_y), button_size_nonpressed)
    #botton PROCESS
    pos_x += 50
    pygame.draw.circle(s_display, black, (pos_x, pos_y), (button_size_nonpressed + 3))
    botton_display_3 = pygame.draw.circle(s_display, dark_sand, (pos_x, pos_y), button_size_nonpressed)

    pos_x = pos_zero_x
    if menu == 'HOME':
        pygame.draw.circle(s_display, black, (pos_x,pos_y), (button_size_pressed + 3))
        botton_display_0 = pygame.draw.circle(s_display, grey1, (pos_x,pos_y), button_size_pressed)
    elif menu == 'CPU':
        pygame.draw.circle(s_display, black, ((pos_x + 50),pos_y), (button_size_pressed + 3))
        botton_display_1 = pygame.draw.circle(s_display, grey1, ((pos_x + 50),pos_y), button_size_pressed)
    elif menu =='FILES':
        pygame.draw.circle(s_display, black, ((pos_x + 100),pos_y), (button_size_pressed + 3))
        botton_display_2 = pygame.draw.circle(s_display, grey1, ((pos_x + 100),pos_y), button_size_pressed)
    elif menu == 'PROCESS':
```

```
        pygame.draw.circle(s_display, black, ((pos_x + 150),pos_y), (button_size_pressed + 3))
        botton_display_3 = pygame.draw.circle(s_display, grey1, ((pos_x + 150),pos_y), button_size_pressed)

    switch_button(connection)

    page_name = font_4.render("Home", 10, dark_sand)
    s_display.blit(page_name, (110, 13))
    page_name = font_4.render("CPU", 10, dark_sand)
    s_display.blit(page_name, (163, 13))
    page_name = font_4.render("Files", 10, dark_sand)
    s_display.blit(page_name, (212, 13))
    page_name = font_4.render("Network", 10, dark_sand)
    s_display.blit(page_name, (255, 13))
    version = font_4.render("Version: 7.0.0", 10, dark_sand)
    s_menu_bottom.blit(version, (710, 20))
    ass = font_4.render("Dev.: Fábio R. P. Nunes", 10, dark_sand)
    s_menu_bottom.blit(ass, (660, 45))
    window.blit(s_menu_bottom_border, (0, WINDOW_HEIGHT))
    window.blit(s_menu_bottom, (5, (WINDOW_HEIGHT + 5)))
    window.blit(s_display, (pos_x_s_display, pos_y_s_display))

    return botton_display_0, botton_display_1, botton_display_2, botton_display_3

def switch_button(connection):
    pos_zero_x = 50
    pos_x = pos_zero_x
    pos_y = ((int(MENU_BOTTOM_BORDER_HEIGHT / 3) + 2) + 13)
    button_size_pressed = 8
    button_size_nonpressed = 10

    pygame.draw.line(s_menu_bottom, black, (pos_x,pos_y), (pos_x+25, pos_y), 20)

    #botton ONLINE
    pygame.draw.circle(s_menu_bottom, black, (pos_x, pos_y), (button_size_nonpressed + 3))
    botton_online = pygame.draw.circle(s_menu_bottom, dark_sand, (pos_x, pos_y), button_size_nonpressed)
    #botton OFFLINE
    pos_x += 25
    pygame.draw.circle(s_menu_bottom, black, (pos_x, pos_y), (button_size_nonpressed + 3))
    botton_offline = pygame.draw.circle(s_menu_bottom, dark_sand, (pos_x, pos_y), button_size_nonpressed)

    pos_x = pos_zero_x
    if connection == 'ONLINE':
        pygame.draw.circle(s_menu_bottom, black, (pos_x,pos_y), (button_size_pressed + 3))
        botton_online = pygame.draw.circle(s_menu_bottom, grey_blue2, (pos_x,pos_y), button_size_pressed)
        status_txt = font_4.render("Online", 10, grey_blue1)
        s_menu_bottom.blit(status_txt, (60, 9))
    elif connection == 'OFFLINE':
        pygame.draw.circle(s_menu_bottom, black, ((pos_x + 25),pos_y), (button_size_pressed + 3))
        botton_offline = pygame.draw.circle(s_menu_bottom, red, ((pos_x + 25),pos_y), button_size_pressed)
        status_txt = font_4.render("Offline", 10, red2)
        s_menu_bottom.blit(status_txt, (60, 9))

    mode_txt = font_4.render("Mode", 10, grey)
    s_menu_bottom.blit(mode_txt, (25, 9))

    return botton_online, botton_offline

def display_next_page_buttom():
    text_button = "Next"
    buttom = pygame.draw.circle(s_header, background, ((WINDOW_WIDTH-50),40), 20)
    pygame.draw.polygon(s_header, black, ((WINDOW_WIDTH-35,45),(WINDOW_WIDTH-55,35),(WINDOW_WIDTH-55,55)))
```

```
    pygame.draw.polygon(s_header, dark_sand, ((WINDOW_WIDTH-40,40),(WINDOW_WIDTH-60,30),(WINDOW_WIDTH-
60,50)))
    s_header.blit(font_3.render(text_button, 10, dark_sand), ((WINDOW_WIDTH-70), 10))
    return buttom

def display_back_page_buttom():
    text_button = "Back"
    buttom = pygame.draw.circle(s_header, background, ((WINDOW_WIDTH-100),40), 20)
    pygame.draw.polygon(s_header, black, ((WINDOW_WIDTH-110,45),(WINDOW_WIDTH-90,35),(WINDOW_WIDTH-90,55)))
    pygame.draw.polygon(s_header, dark_sand, ((WINDOW_WIDTH-115,40),(WINDOW_WIDTH-95,30),(WINDOW_WIDTH-
95,50)))
    s_header.blit(font_3.render(text_button, 10, dark_sand), ((WINDOW_WIDTH-120), 10))
    return buttom

def display_pg_end_buttom():
    text_button = "End"
    buttom = pygame.draw.circle(s_header, background, ((WINDOW_WIDTH-50),40), 20)
    pygame.draw.polygon(s_header, black, ((WINDOW_WIDTH-35,35),(WINDOW_WIDTH-55,35),(WINDOW_WIDTH-45,55)))
    pygame.draw.polygon(s_header, dark_sand, ((WINDOW_WIDTH-40,30),(WINDOW_WIDTH-60,30),(WINDOW_WIDTH-
50,50)))
    pygame.draw.line(s_header, black, (WINDOW_WIDTH-55,55), (WINDOW_WIDTH-35, 55), 3)
    pygame.draw.line(s_header, dark_sand, (WINDOW_WIDTH-60,50), (WINDOW_WIDTH-40, 50), 3)
    s_header.blit(font_3.render(text_button, 10, dark_sand), ((WINDOW_WIDTH-61), 10))
    return buttom

def display_pg_dowm_buttom():
    text_button = "Down"
    buttom = pygame.draw.circle(s_header, background, ((WINDOW_WIDTH-90),40), 20)
    pygame.draw.polygon(s_header, black, ((WINDOW_WIDTH-75,35),(WINDOW_WIDTH-95,35),(WINDOW_WIDTH-85,55)))
    pygame.draw.polygon(s_header, dark_sand, ((WINDOW_WIDTH-80,30),(WINDOW_WIDTH-100,30),(WINDOW_WIDTH-
90,50)))
    s_header.blit(font_3.render(text_button, 10, dark_sand), ((WINDOW_WIDTH-107), 10))
    return buttom

def display_pg_up_buttom():
    text_button = "Up"
    buttom = pygame.draw.circle(s_header, background, ((WINDOW_WIDTH-130),40), 20)
    pygame.draw.polygon(s_header, black, ((WINDOW_WIDTH-115,55),(WINDOW_WIDTH-135,55),(WINDOW_WIDTH-
125,35)))
    pygame.draw.polygon(s_header, dark_sand, ((WINDOW_WIDTH-120,50),(WINDOW_WIDTH-140,50),(WINDOW_WIDTH-
130,30)))
    s_header.blit(font_3.render(text_button, 10, dark_sand), ((WINDOW_WIDTH-138), 10))
    return buttom

def display_pg_top_buttom():
    text_button = "Top"
    buttom = pygame.draw.circle(s_header, background, ((WINDOW_WIDTH-170),40), 20)
    pygame.draw.line(s_header, black, (WINDOW_WIDTH-175,35), (WINDOW_WIDTH-155, 35), 3)
    pygame.draw.polygon(s_header, black, ((WINDOW_WIDTH-155,55),(WINDOW_WIDTH-175,55),(WINDOW_WIDTH-
165,35)))
    pygame.draw.polygon(s_header, dark_sand, ((WINDOW_WIDTH-160,50),(WINDOW_WIDTH-180,50),(WINDOW_WIDTH-
170,30)))
    pygame.draw.line(s_header, dark_sand, (WINDOW_WIDTH-180,30), (WINDOW_WIDTH-160, 30), 3)
    s_header.blit(font_3.render(text_button, 10, dark_sand), ((WINDOW_WIDTH-180), 10))
    return buttom

### Windows Config. ###
WINDOW_WIDTH = 800
WINDOW_HEIGHT = 600
MENU_BOTTOM_BORDER_WIDTH = WINDOW_WIDTH
MENU_BOTTOM_BORDER_HEIGHT = 70
```

```
MENU_BOTTOM_WIDTH = (MENU_BOTTOM_BORDER_WIDTH - 10)
MENU_BOTTOM_HEIGHT = (MENU_BOTTOM_BORDER_HEIGHT - 10)

window = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT + MENU_BOTTOM_BORDER_HEIGHT))

### Surfaces ###
s_menu_bottom = pygame.surface.Surface((MENU_BOTTOM_WIDTH, MENU_BOTTOM_HEIGHT))
s_menu_bottom_border = pygame.surface.Surface((MENU_BOTTOM_BORDER_WIDTH, MENU_BOTTOM_BORDER_HEIGHT))
s_display = pygame.surface.Surface((int(MENU_BOTTOM_WIDTH / 2), MENU_BOTTOM_HEIGHT))
s = pygame.surface.Surface((WINDOW_WIDTH, int((WINDOW_HEIGHT)/4)))
s_header = pygame.surface.Surface((WINDOW_WIDTH-10, 65))
s_pag = pygame.surface.Surface((WINDOW_WIDTH-10, WINDOW_HEIGHT-5))
s_pag_scroll = pygame.surface.Surface((WINDOW_WIDTH-10, 50000))
s_pag_border = pygame.surface.Surface((WINDOW_WIDTH, WINDOW_HEIGHT))
s_info = pygame.surface.Surface(((WINDOW_WIDTH - 10), int((WINDOW_HEIGHT)/4) - 5))
s_cpu = pygame.surface.Surface((WINDOW_WIDTH, int(((WINDOW_HEIGHT)/4)*2)))

### Colors ###
red = (150,0,0)
red2 = (200,0,0)
green = (255,0,0)
blue = (0,0,255)
light_blue = (0,200,255)
black = (0,0,0)
white = (255, 255, 255)
grey = (150, 150, 150)
grey1 = (80, 80, 80)
grey2 = (45, 45, 45)
grey2 = (55, 55, 55)
grey_blue1 = (156, 181, 201)
grey_blue2 = (0, 55, 130)
dark_blue = (0, 0, 50)
sand = (255, 230, 185)
sand2 = (225, 190, 145)
dark_sand = (165, 145, 110)
background = (45, 45, 45)

### Fonts ###
pygame.font.init()
font_1 = pygame.font.SysFont('', 36)
font_2 = pygame.font.SysFont('', 24)
font_3 = pygame.font.SysFont('', 20)
font_4 = pygame.font.SysFont('', 16)

### Menu Settings ###
connection = 'OFFLINE'
menu = 'HOME'
pos_x_s_display = 200
pos_y_s_display = WINDOW_HEIGHT + 5
pos_x_s_header = 5
pos_y_s_header = 0
# Buttons
botton_display_0 = menu_buttons(menu, connection)[0]
botton_display_1 = menu_buttons(menu, connection)[1]
botton_display_2 = menu_buttons(menu, connection)[2]
botton_display_3 = menu_buttons(menu, connection)[3]

botton_online = switch_button(connection)[0]
botton_offline = switch_button(connection)[1]

next_page_buttom = display_next_page_buttom()
```

```
back_page_buttom = display_back_page_buttom()
pg_dowm_buttom = display_pg_dowm_buttom()
pg_up_buttom = display_pg_up_buttom()
pg_end_buttom = display_pg_end_buttom()
pg_top_buttom = display_pg_top_buttom()
```

Arquivo: functions.py

```
import interface

import pygame
import psutil
import os
import datetime
import socket
import subprocess
import nmap

## Network
def get_ip_addresses(family):
    for interface, snics in psutil.net_if_addrs().items():
        for snic in snics:
            if snic.family == family:
                yield (interface, snic.address)

def ping(operational_system_data, hostname):
    args = []
    if operational_system_data == "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 hosts_scan(operational_system_data, subnetwork):
    print("Mapping", subnetwork, "subnetwork...\r")
    hosts_list = []
    return_codes = dict()
    for i in range(1, 255):
        host_test = (subnetwork + '{0}'.format(i))
        return_codes[subnetwork + '{0}'.format(i)] = ping(operational_system_data, host_test)
        if i %20 == 0:
            print(".", end = "")
        if return_codes[subnetwork + '{0}'.format(i)] == 0:
            hosts_list.append(subnetwork + '{0}'.format(i))
    print("\nMapping ready.")
    print("Hosts found:", hosts_list)
    return hosts_list

def ips_scan(hosts_list):
    nm = nmap.PortScanner()
    IPs = dict()
    for i in hosts_list:
        try:
            print("Scanning", i, "IP...")
            nm.scan(i)
            IPs[i] = nm[i]
            print("IP", i,"Scanned.")
        except:
            print("IP", i,"Failed.")
```

```
            continue
    print("Scanning Ready.")
    return IPs

def hosts_info(hosts_data, ips_data):
    IPs = dict()
    IPs_Info = ips_data[0]
    hosts = hosts_data[0]
    for i in hosts:
        IPs[i] = dict()
        try:
            IPs[i]['mac'] = IPs_Info[i]['addresses']['mac']
        except:
            pass
        try:
            IPs[i]['vendor'] = IPs_Info[i]['vendor'][IPs_Info[i]['addresses']['mac']]
        except:
            pass
        try:
            IPs[i]['response'] = IPs_Info[i]['status']['reason']
        except:
            pass
        try:
            IPs[i]['product'] = IPs_Info[i]['product']
            print(IPs[i]['product'])
        except:
            pass
    return IPs

def traffic_interface_scanner():
    io_status = psutil.net_io_counters(pernic=True)
    interface_names = list()
    for i in io_status:
        interface_names.append(str(i))
    interfaces_traffic = dict()
    for j in interface_names:
        interfaces_traffic[j] = dict()
        interfaces_traffic[j]['bytes_sent'] = io_status[j][0]
        interfaces_traffic[j]['bytes_recv'] = io_status[j][1]
        interfaces_traffic[j]['packets_sent'] = io_status[j][2]
        interfaces_traffic[j]['packets_recv'] = io_status[j][3]
    return interfaces_traffic

def connection_check(status):
    if status == 'OK':
        return 'ONLINE'
    elif status == 'ERROR':
        return 'OFFLINE'

## CPU
def cpu_graph_1(connection, cpu_percent_data, cpu_max_freq_data):
    # Data
    if connection == 'ONLINE':
        freq_per_cent = int(cpu_percent_data)
        freq_max = cpu_max_freq_data
    else:
        freq_per_cent = int(psutil.cpu_percent(interval=1))
        freq_max = psutil.cpu_freq().max

    freq_current = ((freq_max * freq_per_cent)/100)
```

```
    # Display
    interface.s.fill(interface.background)

    width_bar_1 = int(interface.WINDOW_WIDTH - 2*20)
    pygame.draw.rect(interface.s, interface.black, (25, 70, width_bar_1, 70))
    pygame.draw.rect(interface.s, interface.grey_blue2, (20, 65, width_bar_1, 70))
    width_bar_2 = int((width_bar_1*freq_per_cent) / 100)
    pygame.draw.rect(interface.s, interface.black, (25, 70, width_bar_2, 70))
    pygame.draw.rect(interface.s, interface.red, (20, 65, width_bar_2, 70))

    pygame.draw.rect(interface.s, interface.black, ((558), (15), (129), (29)))
    pygame.draw.rect(interface.s, interface.background, ((560), (17), (125), (25)))
    pygame.draw.rect(interface.s, interface.black, ((683), (15), (99), (29)))
    pygame.draw.rect(interface.s, interface.background, ((685), (17), (95), (25)))

    pygame.draw.line(interface.s, interface.black, (0,0), ((interface.WINDOW_WIDTH), 0), 8)
    pygame.draw.line(interface.s, interface.black, (0,0), (0, (interface.WINDOW_HEIGHT)), 8)
    pygame.draw.line(interface.s, interface.black, ((interface.WINDOW_WIDTH),0), ((interface.WINDOW_WIDTH),
(interface.WINDOW_HEIGHT)), 12)
    pygame.draw.line(interface.s, interface.black, (15,54), ((interface.WINDOW_WIDTH - 15), 54), 6)

    bar_text_1 = "CPU Usage"
    bar_text_2 = "Current Frequency"
    bar_text_3 = str(freq_current) + " MHz"
    bar_text_inside_1 = str(freq_per_cent) + "%"
    bar_text_inside_2 = str((100 - freq_per_cent)) + "%"
    interface.s.blit(interface.font_1.render(bar_text_1, 10, interface.grey), (20, 18))
    interface.s.blit(interface.font_3.render(bar_text_2, 10, interface.grey), (565, 23))
    interface.s.blit(interface.font_3.render(bar_text_3, 10, interface.dark_sand), (700, 23))
    interface.s.blit(interface.font_2.render(bar_text_inside_1, 10, interface.sand2), (35, 110))
    interface.s.blit(interface.font_2.render(bar_text_inside_2, 10, interface.dark_sand), (730, 110))

    interface.window.blit(interface.s, (0, 0))

def cpu_graph_2(connection, cpu_cores_percent_data):
    # Data
    if connection == 'ONLINE':
        cores_percent = cpu_cores_percent_data
    else:
        cores_percent = psutil.cpu_percent(interval=1, percpu=True)

    cores = len(cores_percent)

    # Display
    interface.s_cpu.fill(interface.background)

    x = y = shift = 10
    d = x + shift
    height = int(interface.s_cpu.get_height() - 2*y)
    width = int((interface.s_cpu.get_width() - 2*y - (cores + 1)*shift ) / cores)
    cores_count = 1
    for i in cores_percent:
        pygame.draw.rect(interface.s_cpu, interface.black, (d+5, y+5, width, height))
        pygame.draw.rect(interface.s_cpu, interface.red, (d, y, width, height))
        pygame.draw.rect(interface.s_cpu, interface.grey_blue2, (d, y, width, ((1-i/100)*height)))

        bar_text_inside = str(int(i)) + "%"
        interface.s_cpu.blit(interface.font_3.render(bar_text_inside, 10, interface.sand2), ((d +
(width/3)), (height - 10)))
        bar_text_inside = "#"+str(cores_count)
        interface.s_cpu.blit(interface.font_3.render(bar_text_inside, 10, interface.dark_sand), ((d +
```

```
(width/3)), y+10))

        cores_count += 1
        d = d + width + shift

    pygame.draw.line(interface.s_cpu, interface.black, (0,0), (0, (interface.WINDOW_HEIGHT)), 8)
    pygame.draw.line(interface.s_cpu, interface.black, ((interface.WINDOW_WIDTH),0),
((interface.WINDOW_WIDTH), (interface.WINDOW_HEIGHT)), 12)
    interface.window.blit(interface.s_cpu, (0, (interface.WINDOW_HEIGHT)/4))

## Memory
def mem_graph(connection, mem_data):
    # Data
    if connection == 'ONLINE':
        mem = mem_data
    else:
        mem = psutil.virtual_memory()

    mem_total = round((mem.total / (1024*1024*1024)), 2)
    mem_used = round((mem.used / (1024*1024*1024)), 2)
    mem_free = round((mem.free / (1024*1024*1024)), 2)
    mem_percent = int(mem.percent)

    # Display
    interface.s.fill(interface.background)

    width_bar_1 = int(interface.WINDOW_WIDTH - 2*20)
    pygame.draw.rect(interface.s, interface.black, (25, 70, width_bar_1, 70))
    pygame.draw.rect(interface.s, interface.grey_blue2, (20, 65, width_bar_1, 70))
    width_bar_2 = int((width_bar_1*mem.percent) / 100)
    pygame.draw.rect(interface.s, interface.black, (25, 70, width_bar_2, 70))
    pygame.draw.rect(interface.s, interface.red, (20, 65, width_bar_2, 70))

    pygame.draw.rect(interface.s, interface.black, ((643), (10), (49), (29)))
    pygame.draw.rect(interface.s, interface.background, ((645), (12), (45), (25)))
    pygame.draw.rect(interface.s, interface.black, ((688), (10), (79), (29)))
    pygame.draw.rect(interface.s, interface.background, ((690), (12), (75), (25)))

    pygame.draw.line(interface.s, interface.black, (0,0), ((interface.WINDOW_WIDTH), 0), 8)
    pygame.draw.line(interface.s, interface.black, (0,0), (0, (interface.WINDOW_HEIGHT)), 8)
    pygame.draw.line(interface.s, interface.black, ((interface.WINDOW_WIDTH),0), ((interface.WINDOW_WIDTH),
(interface.WINDOW_HEIGHT)), 12)
    pygame.draw.line(interface.s, interface.black, (15,45), ((interface.WINDOW_WIDTH -15), 45), 6)

    bar_text_1 = "Memory Usage"
    bar_text_2 = "Total"
    bar_text_3 = str(mem_total) + " GB"
    bar_text_inside_1 = str(mem_used) + " Used"
    bar_text_inside_2 = str(mem_free) + " Free"
    bar_text_inside_3 = str(mem_percent) + "%"
    bar_text_inside_4 = str(100 - mem_percent) + "%"
    interface.s.blit(interface.font_1.render(bar_text_1, 10, interface.grey), (20, 13))
    interface.s.blit(interface.font_3.render(bar_text_2, 10, interface.grey), (653, 18))
    interface.s.blit(interface.font_3.render(bar_text_3, 10, interface.dark_sand), (700, 18))
    interface.s.blit(interface.font_2.render(bar_text_inside_1, 10, interface.sand2), (35, 110))
    interface.s.blit(interface.font_2.render(bar_text_inside_2, 10, interface.dark_sand), (675, 110))
    interface.s.blit(interface.font_2.render(bar_text_inside_3, 10, interface.sand2), (35, 75))
    interface.s.blit(interface.font_2.render(bar_text_inside_4, 10, interface.dark_sand), (730, 75))

    interface.window.blit(interface.s, (0, ((interface.WINDOW_HEIGHT)/4)))
```

```
## HD
def hd_graph(connection, hd_data):
    # Data
    if connection == 'ONLINE':
        hd = hd_data
    else:
        hd = hd_data

    hd_total = round((hd.total / (1024*1024*1024)), 2)
    hd_used = round((hd.used / (1024*1024*1024)), 2)
    hd_free = round((hd.free / (1024*1024*1024)), 2)
    hd_percent = int(hd.percent)

    # Display
    interface.s.fill(interface.background)

    width_bar_1 = int(interface.WINDOW_WIDTH - 2*20)
    pygame.draw.rect(interface.s, interface.black, (25, 70, width_bar_1, 70))
    pygame.draw.rect(interface.s, interface.grey_blue2, (20, 65, width_bar_1, 70))
    width_bar_2 = int((width_bar_1*hd.percent) / 100)
    pygame.draw.rect(interface.s, interface.black, (25, 70, width_bar_2, 70))
    pygame.draw.rect(interface.s, interface.red, (20, 65, width_bar_2, 70))

    pygame.draw.rect(interface.s, interface.black, ((643), (10), (49), (29)))
    pygame.draw.rect(interface.s, interface.background, ((645), (12), (45), (25)))
    pygame.draw.rect(interface.s, interface.black, ((688), (10), (84), (29)))
    pygame.draw.rect(interface.s, interface.background, ((690), (12), (80), (25)))

    pygame.draw.line(interface.s, interface.black, (0,0), ((interface.WINDOW_WIDTH), 0), 8)
    pygame.draw.line(interface.s, interface.black, (0,0), (0, (interface.WINDOW_HEIGHT)), 8)
    pygame.draw.line(interface.s, interface.black, ((interface.WINDOW_WIDTH),0), ((interface.WINDOW_WIDTH),
(interface.WINDOW_HEIGHT)), 12)
    pygame.draw.line(interface.s, interface.black, (15,45), ((interface.WINDOW_WIDTH -15), 45), 6)

    bar_text_1 = "Primary Storage Usage"
    bar_text_2 = "Total"
    bar_text_3 = str(hd_total) + " GB"
    bar_text_inside_01 = str(hd_used) + " Used"
    bar_text_inside_02 = str(hd_free) + " Free"
    bar_text_inside_03 = str(hd_percent) + "%"
    bar_text_inside_04 = str(100 - hd_percent) + "%"
    interface.s.blit(interface.font_1.render(bar_text_1, 10, interface.grey), (20, 13))
    interface.s.blit(interface.font_3.render(bar_text_2, 10, interface.grey), (653, 18))
    interface.s.blit(interface.font_3.render(bar_text_3, 10, interface.dark_sand), (700, 18))
    interface.s.blit(interface.font_2.render(bar_text_inside_01, 10, interface.sand2), (35, 110))
    interface.s.blit(interface.font_2.render(bar_text_inside_02, 10, interface.dark_sand), (675, 110))
    interface.s.blit(interface.font_2.render(bar_text_inside_03, 10, interface.sand2), (35, 75))
    interface.s.blit(interface.font_2.render(bar_text_inside_04, 10, interface.dark_sand), (730, 75))

    interface.window.blit(interface.s, (0, ((((interface.WINDOW_HEIGHT)/4)*2))))

## Informations
def pc_cpu_inf(pc_name_data, operational_system_data, os_version_data, cpu_data, arc_data, p_cores_data,
l_cores_data):
    # Data
    inf = cpu_data
    cpu_name = inf['brand_raw']
    cpu_arch = inf['arch']
    cpu_bits = inf['bits']
    p_cores = p_cores_data
    l_cores = l_cores_data
```

```
    # Diplay
    interface.s_info.fill(interface.background)
    interface.s.fill(interface.black)

    text = "PC & CPU Information"
    interface.s_info.blit(interface.font_1.render(text, 10, interface.grey), (20, 10))
    pygame.draw.line(interface.s_info, interface.black, (10,45), ((interface.WINDOW_WIDTH -20), 45), 6)

    pos_zero_x = 20
    pos_zero_y = 55
    box_zero_size_x = 73
    box_zero_size_y = 15
    pos_x = pos_zero_x
    pos_y = pos_zero_y
    box_size_x = box_zero_size_x
    box_size_y = box_zero_size_y

    box_size_x = 65
    text = "PC Name"
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.dark_sand), ((pos_x + 5), (pos_y +
3)))
    pos_x += box_size_x
    text = str(pc_name_data)
    box_size_x = 213
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.grey), ((pos_x + 5), (pos_y + 3)))
    pos_x = pos_zero_x

    pos_y += 17
    box_size_x = 110
    text = "Operating System"
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.dark_sand), ((pos_x + 5), (pos_y +
3)))
    pos_x += box_size_x
    text = str(operational_system_data) + " " + str(os_version_data)
    box_size_x = 175
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.grey), ((pos_x + 5), (pos_y + 3)))
```

```
    pos_x = pos_zero_x

    pos_y += 17
    box_size_x = box_zero_size_x
    text = "CPU Name"
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.dark_sand), ((pos_x + 5), (pos_y +
3)))
    pos_x += box_size_x
    text = str(cpu_name)
    box_size_x = 225
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.grey), ((pos_x + 5), (pos_y + 3)))
    pos_x = pos_zero_x

    pos_y += 17
    box_size_x = 80
    text = "Architecture"
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.dark_sand), ((pos_x + 5), (pos_y +
3)))
    pos_x += box_size_x
    text = str(cpu_arch) + " (" + str(arc_data) + ")"
    box_size_x = 105
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.grey), ((pos_x + 5), (pos_y + 3)))
    pos_x = pos_zero_x

    pos_y += 17
    box_size_x = 85
    text = "Word Length"
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.dark_sand), ((pos_x + 5), (pos_y +
3)))
    pos_x += box_size_x
    text = str(cpu_bits) + " bits"
```

```
    box_size_x = 65
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.grey), ((pos_x + 5), (pos_y + 3)))
    pos_x = (pos_zero_x + 400)

    pos_y = pos_zero_y
    box_size_x = 45
    text = "Cores"
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.dark_sand), ((pos_x + 5), (pos_y +
3)))
    pos_y += 17
    box_size_x = 60
    text = "Physical"
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.dark_sand), ((pos_x + 5), (pos_y +
3)))
    pos_x += box_size_x
    text = str(p_cores)
    box_size_x = 20
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.grey), ((pos_x + 5), (pos_y + 3)))
    pos_x = (pos_zero_x + 400)
    pos_y += 17
    box_size_x = 55
    text = "Logical"
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.dark_sand), ((pos_x + 5), (pos_y +
3)))
    pos_x += box_size_x
    text = str(l_cores)
    box_size_x = 25
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x + 1), (pos_y + 1), (box_size_x + 4),
(box_size_y + 4)))
    pygame.draw.rect(interface.s_info, interface.black, ((pos_x - 2), (pos_y - 2), (box_size_x + 4),
(box_size_y + 4)))
```

```
    pygame.draw.rect(interface.s_info, interface.background, ((pos_x), (pos_y), (box_size_x),
(box_size_y)))
    interface.s_info.blit(interface.font_4.render(text, 10, interface.grey), ((pos_x + 5), (pos_y + 3)))
    pos_x = (pos_zero_x + 400)

    interface.window.blit(interface.s, (0, ((((interface.WINDOW_HEIGHT)/4)*3))))
    interface.window.blit(interface.s_info, (5, (((((interface.WINDOW_HEIGHT)/4)*3)+5))))

## Files Page
def files_page_header(local_data, files_page_number_data, files_page_menu):
    interface.s_header.fill(interface.background)

    if (((files_page_number_data) > (1)) and ((int(files_page_menu)) < (files_page_number_data))):
        interface.display_next_page_buttom()
    if files_page_menu != '1':
        interface.display_back_page_buttom()

    pos_x = 20
    pos_y = 10
    text_0 = "Local Files"
    interface.s_header.blit(interface.font_1.render(text_0, 10, interface.grey), (pos_x, pos_y))
    pos_y += 30
    text_1 = str(local_data)
    interface.s_header.blit(interface.font_3.render(text_1, 10, interface.grey), (pos_x, pos_y))
    pos_y += 28

    pygame.draw.line(interface.s_header, interface.black, (0,0), ((interface.WINDOW_WIDTH), 0), 8)
    pygame.draw.line(interface.s_header, interface.black, (10,60), ((interface.WINDOW_WIDTH -20), 60), 6)

    interface.window.blit(interface.s_header, (interface.pos_x_s_header, interface.pos_y_s_header))

def scan_files(local_data):
    lista = os.listdir(local_data)
    files = dict()
    for i in lista:
        filepath = os.path.join(local_data, i)
        if os.path.isfile(filepath):
            files[i] = dict()
            files[i]['Size'] = os.stat(filepath).st_size
            files[i]['Created'] =
datetime.datetime.fromtimestamp(os.stat(filepath).st_ctime).strftime("%d-%m-%Y %H:%M:%S")
            files[i]['Mod'] =
datetime.datetime.fromtimestamp(os.stat(filepath).st_mtime).strftime("%d-%m-%Y %H:%M:%S")
    return files

def create_files_pages(files_data):
    files = files_data
    files_count = 1
    page_number = 1
    pages = dict()
    pages[page_number] = dict()
    for i in list(files):
        pages[page_number][i] = dict()
        pages[page_number][i]['Size'] = files[i]['Size']
        pages[page_number][i]['Created'] = files[i]['Created']
        pages[page_number][i]['Mod'] = files[i]['Mod']
        files_count += 1
        if files_count > 12:
            page_number += 1
            pages[page_number] = dict()
            files_count = 1
```

```
    return pages

def files_pages_count(files_pages_data):
    return len(files_pages_data)

def files_display(files_pages_data, files_page_menu):
    interface.s_pag.fill(interface.background)
    interface.s_pag_border.fill(interface.black)

    files = files_pages_data[int(files_page_menu)]

    pos_x_files = 47
    pos_y_files = 78
    files_count = 0
    for i in list(files):
        file_text = '{:^0.34}'.format(str(i))
        file_size_text = "Size: " + '{:^0.38}'.format(str(files[i]['Size']) + " bytes")
        file_created_text = "Created: " + '{:^0.38}'.format(str(files[i]['Created']))
        file_mod_text = "Modification: "+ '{:^0.38}'.format(str(files[i]['Mod']))

        pygame.draw.rect(interface.s_pag, interface.black, (pos_x_files, pos_y_files+3,
int((interface.WINDOW_WIDTH)/2)-75, (73)))
        pygame.draw.rect(interface.s_pag, interface.black, (pos_x_files-5, pos_y_files-4,
int((interface.WINDOW_WIDTH)/2)-75, (22)))
        pygame.draw.rect(interface.s_pag, interface.grey2, (pos_x_files-4, pos_y_files-3,
int((interface.WINDOW_WIDTH)/2)-77, (20)))
        interface.s_pag.blit(interface.font_3.render(file_text, 10, interface.dark_sand), (pos_x_files + 8,
pos_y_files))
        pygame.draw.circle(interface.s_pag, interface.black, ((pos_x_files-5),(pos_y_files + 8)), 8)
        pygame.draw.circle(interface.s_pag, interface.grey2, ((pos_x_files-5),(pos_y_files + 8)), 6)
        pos_y_files += 18
        pygame.draw.rect(interface.s_pag, interface.black, (pos_x_files-5, pos_y_files,
int((interface.WINDOW_WIDTH)/2)-75, (53)))
        pygame.draw.rect(interface.s_pag, interface.grey2, (pos_x_files-4, pos_y_files+1,
int((interface.WINDOW_WIDTH)/2)-77, (51)))
        pos_y_files += 5
        interface.s_pag.blit(interface.font_4.render(file_size_text, 10, interface.dark_sand),
(pos_x_files, pos_y_files))
        pos_y_files += 15
        interface.s_pag.blit(interface.font_4.render(file_created_text, 10, interface.dark_sand),
(pos_x_files, pos_y_files))
        pos_y_files += 15
        interface.s_pag.blit(interface.font_4.render(file_mod_text, 10, interface.dark_sand), (pos_x_files,
pos_y_files))
        pos_y_files += 30

        files_count += 1
        if (files_count == 6):
            pos_x_files += int(((interface.WINDOW_WIDTH)/2)-22)
            pos_y_files = 78

    interface.window.blit(interface.s_pag_border, (0, 0))
    interface.window.blit(interface.s_pag, (5, 5))

#Processes
def scann_process_data():
    processes = psutil.pids()
    process_data = dict()
    for i in processes:
        try:
            p = psutil.Process(i)
```

```
            proc_name = str(p.name())
            if proc_name not in process_data:
                process_data[proc_name] = dict()
                process_data[proc_name]['Status'] = list()
                process_data[proc_name]['PIDs'] = list()
                process_data[proc_name]['Create Time'] = list()
                process_data[proc_name]['Connections'] = list()
            process_data[proc_name]['Status'].append(p.status())
            process_data[proc_name]['PIDs'].append(i)
            process_data[proc_name]['Create
Time'].append(datetime.datetime.fromtimestamp(p.create_time()).strftime("%d-%m-%Y %H:%M:%S"))
            if p.connections() != []:
                process_data[proc_name]['Connections'].append(p.connections())
            else:
                process_data[proc_name]['Connections'].append('OFFLINE')
        except (psutil.AccessDenied, psutil.ZombieProcess, psutil.NoSuchProcess):
            continue

    all_process_data = dict()
    for i in process_data:
        all_process_data[i] = dict()
        all_process_data[i]['Qty PIDs'] = len(process_data[i]['PIDs'])
        all_process_data[i]['PIDs List'] = process_data[i]['PIDs']
        for j in range(all_process_data[i]['Qty PIDs']):
            all_process_data[i][process_data[i]['PIDs'][j]] = dict()
            all_process_data[i][process_data[i]['PIDs'][j]]['#'] = process_data[i]['PIDs'][j]
            all_process_data[i][process_data[i]['PIDs'][j]]['Status'] = process_data[i]['Status'][j]
            all_process_data[i][process_data[i]['PIDs'][j]]['Create Time'] = process_data[i]['Create
Time'][j]
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections'] = dict()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Connections Status'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Address Family'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Packets Type'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['LIP'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['LPort'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['RIP'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['RPort'] = list()
            if process_data[i]['Connections'][j] != 'OFFLINE':
                for c in process_data[i]['Connections'][j]:
                    all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Connections
Status'].append(c.status)
                    if c.family == socket.AF_INET:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Address
Family'].append('IPv4')
                    elif c.family == socket.AF_INET6:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Address
Family'].append('IPv6')
                    elif c.family == socket.AF_UNIX:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Address
Family'].append('Unix')
                    else:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Address
Family'].append('-')

                    if c.type == socket.SOCK_STREAM:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Packets
Type'].append('TCP')
                    elif c.type == socket.SOCK_DGRAM:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Packets
Type'].append('UDP')
                    elif c.type == socket.SOCK_RAW:
```

```
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Packets
Type'].append('IP')
                    else:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Packets
Type'].append('-')

                    d_count = 1
                    for d in c.laddr:
                        if d_count == 1:
                            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['LIP'].append(d)
                            d_count += 1
                        elif d_count == 2:

all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['LPort'].append(d)
                            d_count = 1
                    for d in c.raddr:
                        if d_count == 1:
                            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['RIP'].append(d)
                            d_count += 1
                        elif d_count == 2:

all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['RPort'].append(d)
                            d_count = 1
            else:
                all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Connections Status'] =
process_data[i]['Connections'][j]
    return all_process_data

# Display Network and Processes
def network_and_processes_page_header(ipv4s_data, mac_address_data, processes_page_menu):
    interface.s_header.fill(interface.background)

    interface.display_pg_dowm_buttom()
    interface.display_pg_up_buttom()
    interface.display_pg_end_buttom()
    interface.display_pg_top_buttom()

    pos_x = 20
    pos_y = 10
    text_0 = "Network & Process Information"
    interface.s_header.blit(interface.font_1.render(text_0, 10, interface.grey), (pos_x, pos_y))
    pos_y += 30
    for i in range(len(ipv4s_data)):
        if (ipv4s_data[i][0] != 'Loopback Pseudo-Interface 1') and (ipv4s_data[i][0] != 'lo0'):
            text = ipv4s_data[i][0] + ": " + ipv4s_data[i][1]
            interface.s_header.blit(interface.font_3.render(text, 10, interface.dark_sand), (pos_x, pos_y))
            pos_x += 150
    text = "MAC Address: " + str(mac_address_data[0][1])
    interface.s_header.blit(interface.font_3.render(text, 10, interface.dark_sand), (pos_x, pos_y))

    pygame.draw.line(interface.s_header, interface.black, (0,0), ((interface.WINDOW_WIDTH), 0), 8)
    pygame.draw.line(interface.s_header, interface.black, (10,60), ((interface.WINDOW_WIDTH -20), 60), 6)

    interface.window.blit(interface.s_header, (interface.pos_x_s_header, interface.pos_y_s_header))

def network_and_processes_display(connection, hosts_data, ips_data, processes_data, traffic_interface_data,
processes_page_scroll_y, processes_page_menu):
    interface.s_pag_scroll.fill(interface.background)
    interface.s_pag_border.fill(interface.black)

    process_data = processes_data
```

```
    pos_zero_x_process = 450
    pos_zero_y_process = 75
    pos_x_process = pos_zero_x_process
    pos_y_process = pos_zero_y_process
    box_name_size_x = 250
    box_name_size_y = 20
    box_size_x = 46
    box_size_y = 19
    for i in process_data:
        pos_x_process = pos_zero_x_process

        Name_txt = str(i)
        pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3), (pos_y_process +
3), (box_name_size_x + 2), (box_name_size_y + 2)))
        pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1), (pos_y_process -
1), (box_name_size_x + 2), (box_name_size_y + 2)))
        pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process), (pos_y_process),
box_name_size_x, box_name_size_y))
        interface.s_pag_scroll.blit(interface.font_3.render(Name_txt, 10, interface.dark_sand),
((pos_x_process + 10), (pos_y_process + 3)))

        pos_x_ball = pos_x_process
        pos_y_ball = pos_y_process
        pos_y_process += 21
        Head_txt = "Qty of PIDs:"
        box_size_x = 100
        pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3), (pos_y_process +
3), (box_size_x + 2), (box_size_y + 2)))
        pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1), (pos_y_process -
1), (box_size_x + 2), (box_size_y + 2)))
        pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process), (pos_y_process),
box_size_x, box_size_y))
        interface.s_pag_scroll.blit(interface.font_4.render(Head_txt, 10, interface.dark_sand),
((pos_x_process + 10), (pos_y_process + 5)))
        pos_x_process += box_size_x

        Resut_txt = str(process_data[i]['Qty PIDs'])
        box_size_x = 55
        pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3), (pos_y_process +
3), (box_size_x + 2), (box_size_y + 2)))
        pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1), (pos_y_process -
1), (box_size_x + 2), (box_size_y + 2)))
        pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process), (pos_y_process),
box_size_x, box_size_y))
        interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10, interface.grey), ((pos_x_process
+ 10), (pos_y_process + 5)))
        pos_x_process = pos_zero_x_process

        ### PIDs Horizontal Stats
        for f in process_data[i]['PIDs List']:
            pos_x_process = pos_zero_x_process
            pos_y_process += 20

            Head_txt = "PID"
            box_size_x = 55
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3), (pos_y_process
+ 3), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1), (pos_y_process
- 1), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process), (pos_y_process),
```

```
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(Head_txt, 10, interface.dark_sand),
((pos_x_process + 10), (pos_y_process + 5)))
            pos_y_process += 20

            Resut_txt = str(process_data[i][f]['#'])
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3), (pos_y_process
+ 3), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1), (pos_y_process
- 1), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process), (pos_y_process),
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10, interface.grey),
((pos_x_process + 10), (pos_y_process + 5)))
            pos_x_process += box_size_x
            pos_y_process -= 20

            Head_txt = "Status"
            box_size_x = 73
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3), (pos_y_process
+ 3), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1), (pos_y_process
- 1), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process), (pos_y_process),
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(Head_txt, 10, interface.dark_sand),
((pos_x_process + 10), (pos_y_process + 5)))
            pos_y_process += 20

            Resut_txt = str(process_data[i][f]['Status'])
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3), (pos_y_process
+ 3), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1), (pos_y_process
- 1), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process), (pos_y_process),
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10, interface.grey),
((pos_x_process + 10), (pos_y_process + 5)))
            pos_x_process += box_size_x
            pos_y_process -= 20

            Head_txt = "Create Time"
            box_size_x = 152
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3), (pos_y_process
+ 3), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1), (pos_y_process
- 1), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process), (pos_y_process),
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(Head_txt, 10, interface.dark_sand),
((pos_x_process + 10), (pos_y_process + 5)))
            pos_y_process += 20

            Resut_txt = str(process_data[i][f]['Create Time'])
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3), (pos_y_process
+ 3), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1), (pos_y_process
- 1), (box_size_x + 2), (box_size_y + 2)))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process), (pos_y_process),
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10, interface.grey),
```

```
((pos_x_process + 10), (pos_y_process + 5)))

            if process_data[i][f]['Connections']['Connections Status'] != 'OFFLINE':
                pygame.draw.circle(interface.s_pag_scroll, interface.black, ((pos_x_ball),(pos_y_ball +
10)), 8)
                pygame.draw.circle(interface.s_pag_scroll, interface.grey_blue2, ((pos_x_ball),(pos_y_ball
+ 10)), 6)

                for d in range(len(process_data[i][f]['Connections']['Connections Status'])):
                    pos_x_process = pos_zero_x_process
                    pos_y_process += 20
                    Head_txt = "Connection:"
                    box_size_x = 90
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                    interface.s_pag_scroll.blit(interface.font_4.render(Head_txt, 10, interface.dark_sand),
((pos_x_process + 10), (pos_y_process + 5)))
                    pos_x_process += box_size_x

                    Resut_txt = str(process_data[i][f]['Connections']['Connections Status'][d])
                    box_size_x = 100
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                    interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10, interface.grey),
((pos_x_process + 10), (pos_y_process + 5)))
                    pos_x_process += box_size_x

                    Resut_txt = str(process_data[i][f]['Connections']['Address Family'][d])
                    box_size_x = 50
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                    interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10, interface.grey),
((pos_x_process + 10), (pos_y_process + 5)))
                    pos_x_process += box_size_x

                    Resut_txt = str(process_data[i][f]['Connections']['Packets Type'][d])
                    box_size_x = 40
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                    interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10, interface.grey),
((pos_x_process + 10), (pos_y_process + 5)))
                    pos_y_process += 20
                    pos_x_process = pos_zero_x_process
```

```
                    Head_txt = "Local IP:"
                    box_size_x = 90
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                    interface.s_pag_scroll.blit(interface.font_4.render(Head_txt, 10, interface.dark_sand),
((pos_x_process + 10), (pos_y_process + 5)))
                    pos_x_process += box_size_x

                    Resut_txt = '{:.15}'.format(str(process_data[i][f]['Connections']['LIP'][d]))
                    box_size_x = 100
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                    interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10, interface.grey),
((pos_x_process + 10), (pos_y_process + 5)))
                    pos_x_process += box_size_x

                    Resut_txt = str(process_data[i][f]['Connections']['LPort'][d])
                    box_size_x = 90
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                    interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10, interface.grey),
((pos_x_process + 10), (pos_y_process + 5)))
                    pos_x_process = pos_zero_x_process

                    if process_data[i][f]['Connections']['RIP'] != []:
                        pos_y_process += 20
                        try:
                            Head_txt = "Remote IP:"
                            box_size_x = 90
                            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                            interface.s_pag_scroll.blit(interface.font_4.render(Head_txt, 10,
interface.dark_sand), ((pos_x_process + 10), (pos_y_process + 5)))
                            pos_x_process += box_size_x

                            Resut_txt = '{:.15}'.format(str(process_data[i][f]['Connections']['RIP'][d]))
                            box_size_x = 100
                            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                            interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10,
```

```
interface.grey), ((pos_x_process + 10), (pos_y_process + 5)))
                            pos_x_process += box_size_x

                            Resut_txt = str(process_data[i][f]['Connections']['RPort'][d])
                            box_size_x = 90
                            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                            pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                            interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10,
interface.grey), ((pos_x_process + 10), (pos_y_process + 5)))
                            pos_x_process = pos_zero_x_process
                        except:
                            continue
            else:
                pygame.draw.circle(interface.s_pag_scroll, interface.black, ((pos_x_ball),(pos_y_ball +
10)), 8)
                pygame.draw.circle(interface.s_pag_scroll, interface.red, ((pos_x_ball),(pos_y_ball + 10)),
6)

                pos_x_process = pos_zero_x_process
                pos_y_process += 20

                Head_txt = "Connection:"
                box_size_x = 90
                pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                interface.s_pag_scroll.blit(interface.font_4.render(Head_txt, 10, interface.dark_sand),
((pos_x_process + 10), (pos_y_process + 5)))
                pos_x_process += box_size_x

                Resut_txt = str(process_data[i][f]['Connections']['Connections Status'])
                box_size_x = 100
                pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process + 3),
(pos_y_process + 3), (box_size_x + 2), (box_size_y + 2)))
                pygame.draw.rect(interface.s_pag_scroll, interface.black, ((pos_x_process - 1),
(pos_y_process - 1), (box_size_x + 2), (box_size_y + 2)))
                pygame.draw.rect(interface.s_pag_scroll, interface.grey2, ((pos_x_process),
(pos_y_process), box_size_x, box_size_y))
                interface.s_pag_scroll.blit(interface.font_4.render(Resut_txt, 10, interface.grey),
((pos_x_process + 10), (pos_y_process + 5)))
        pos_y_process += 30

    if connection == 'ONLINE':
        traffic = traffic_interface_data
    else:
        traffic = traffic_interface_scanner()

    box_size_x = 43
    box_size_y = 19
    pos_x_traffic = 47
    pos_y_traffic = 78
    for i in list(traffic):
```

```
        if (str(i) != 'Loopback Pseudo-Interface 1'):
            text = str(i)
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic-1,
int((interface.WINDOW_WIDTH)/2)-255, (22)))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic-4,
int((interface.WINDOW_WIDTH)/2)-255, (22)))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic-3,
int((interface.WINDOW_WIDTH)/2)-257, (20)))
            interface.s_pag_scroll.blit(interface.font_3.render(text, 10, interface.dark_sand),
(pos_x_traffic + 8, pos_y_traffic))

            pygame.draw.polygon(interface.s_pag_scroll, interface.black, (((pos_x_traffic-
4),(pos_y_traffic+1)), (pos_x_traffic + 3, pos_y_traffic + 6), (pos_x_traffic-4, pos_y_traffic + 11)))

            pos_y_traffic += 18
            text = "Bytes"
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic+3,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic+1,
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.dark_sand),
(pos_x_traffic + 3, pos_y_traffic + 5))

            pos_x_traffic += box_size_x + 2
            text = "Sent"
            box_size_x -= 3
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic+3,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic+1,
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.dark_sand),
(pos_x_traffic + 3, pos_y_traffic + 5))

            pos_x_traffic += box_size_x + 2
            box_size_x += 50
            text = '{:^0.38}'.format(str(traffic[i]['bytes_sent']))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic+3,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic+1,
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.grey), (pos_x_traffic +
3, pos_y_traffic + 5))

            pos_x_traffic += box_size_x + 2
            box_size_x -= 25
            text = "Received"
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic+3,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic+1,
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.dark_sand),
(pos_x_traffic + 3, pos_y_traffic + 5))
```

```
            pos_x_traffic += box_size_x + 2
            box_size_x += 30
            text = '{:^0.37}'.format(str(traffic[i]['bytes_recv']))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic+3,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic+1,
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.grey), (pos_x_traffic +
3, pos_y_traffic + 5))

            pos_x_traffic = 47
            pos_y_traffic += 21
            box_size_x -= 40
            text = "Packets"
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic+3,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic+1,
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.dark_sand),
(pos_x_traffic + 3, pos_y_traffic + 5))

            pos_x_traffic += box_size_x + 2
            box_size_x -= 15
            text = "Sent"
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic+3,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic+1,
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.dark_sand),
(pos_x_traffic+3, pos_y_traffic + 5))

            pos_x_traffic += box_size_x + 2
            box_size_x += 38
            text = '{:^0.38}'.format(str(traffic[i]['packets_sent']))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic+3,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic+1,
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.grey), (pos_x_traffic +
3, pos_y_traffic + 5))

            pos_x_traffic += box_size_x + 2
            box_size_x -= 13
            text = "Received"
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic+3,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic+1,
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.dark_sand),
(pos_x_traffic + 3, pos_y_traffic + 5))
```

```
            pos_x_traffic += box_size_x + 2
            box_size_x += 30
            text = '{:^0.38}'.format(str(traffic[i]['packets_sent']))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2, pos_y_traffic+3,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5, pos_y_traffic,
box_size_x+2, box_size_y+2))
            pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4, pos_y_traffic+1,
box_size_x, box_size_y))
            interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.grey), (pos_x_traffic +
3, pos_y_traffic + 5))

            pos_x_traffic = 47
            pos_y_traffic += 50

    text = "Network Devices"
    interface.s_pag_scroll.blit(interface.font_2.render(text, 10, interface.grey), (pos_x_traffic - 15,
pos_y_traffic))
    pos_y_traffic += 30
    pygame.draw.line(interface.s_pag_scroll, interface.black, (20, pos_y_traffic), (400, pos_y_traffic), 6)

    if (hosts_data != []) and (ips_data != []):
        IPs = hosts_info(hosts_data, ips_data)

        pos_y_traffic += 30
        box_size_x = 40
        box_size_y = 19
        for i in list(IPs):
            try:
                if (IPs[i]['response'] != 'localhost-response'):
                    pos_x_traffic = 47
                    text = str(i)
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2,
pos_y_traffic-1, int((interface.WINDOW_WIDTH)/2)-255, (22)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5,
pos_y_traffic-4, int((interface.WINDOW_WIDTH)/2)-255, (22)))
                    pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4,
pos_y_traffic-3, int((interface.WINDOW_WIDTH)/2)-257, (20)))
                    interface.s_pag_scroll.blit(interface.font_3.render(text, 10, interface.dark_sand),
(pos_x_traffic + 8, pos_y_traffic))

                    pygame.draw.line(interface.s_pag_scroll, interface.black, (pos_x_traffic -
17,pos_y_traffic + 6), (pos_x_traffic + 3, pos_y_traffic + 6), 9)
                    pygame.draw.line(interface.s_pag_scroll, interface.grey_blue2, (pos_x_traffic -
15,pos_y_traffic + 5), (pos_x_traffic + 1, pos_y_traffic + 5), 6)

                    try:
                        pos_x_traffic = 47
                        pos_y_traffic += 18
                        box_size_x = 40
                        text = "MAC"
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2,
pos_y_traffic+3, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5,
pos_y_traffic, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4,
pos_y_traffic+1, box_size_x, box_size_y))
                        interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.dark_sand),
(pos_x_traffic + 3, pos_y_traffic + 5))
                        pos_x_traffic += box_size_x + 1
                        text = '{:^0.38}'.format(str(IPs[i]['mac']))
```

```
                        box_size_x = 115
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2,
pos_y_traffic+3, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5,
pos_y_traffic, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4,
pos_y_traffic+1, box_size_x, box_size_y))
                        interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.grey),
(pos_x_traffic + 3, pos_y_traffic + 5))
                    except:
                        pass
                    try:
                        pos_x_traffic = 47
                        pos_y_traffic += 19
                        box_size_x = 55
                        text = "Vendor"
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2,
pos_y_traffic+3, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5,
pos_y_traffic, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4,
pos_y_traffic+1, box_size_x, box_size_y))
                        interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.dark_sand),
(pos_x_traffic + 3, pos_y_traffic + 5))
                        pos_x_traffic += box_size_x + 1
                        text = '{:.37}'.format(str(IPs[i]['vendor']))
                        box_size_x = 225
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2,
pos_y_traffic+3, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5,
pos_y_traffic, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4,
pos_y_traffic+1, box_size_x, box_size_y))
                        interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.grey),
(pos_x_traffic + 3, pos_y_traffic + 5))
                    except:
                        pass
                    try:
                        pos_x_traffic = 47
                        pos_y_traffic += 20
                        box_size_x = 75
                        text = "Rensponse"
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2,
pos_y_traffic+3, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5,
pos_y_traffic, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4,
pos_y_traffic+1, box_size_x, box_size_y))
                        interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.dark_sand),
(pos_x_traffic + 3, pos_y_traffic + 5))
                        pos_x_traffic += box_size_x + 1
                        text = '{:.38}'.format(str(IPs[i]['response']))
                        box_size_x = 150
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-2,
pos_y_traffic+3, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.black, (pos_x_traffic-5,
pos_y_traffic, box_size_x+2, box_size_y+2))
                        pygame.draw.rect(interface.s_pag_scroll, interface.grey2, (pos_x_traffic-4,
pos_y_traffic+1, box_size_x, box_size_y))
                        interface.s_pag_scroll.blit(interface.font_4.render(text, 10, interface.grey),
(pos_x_traffic + 3, pos_y_traffic + 5))
```

```
                        pos_y_traffic += 35
                    except:
                        pass
            except:
                continue
    else:
        mapping_text = "Mapping Network..."
        interface.s_pag_scroll.blit(interface.font_1.render(mapping_text, 10, interface.dark_sand), (100,
pos_y_traffic + 100))

    interface.window.blit(interface.s_pag_border, (0, 0))
    interface.window.blit(interface.s_pag_scroll, (5, processes_page_scroll_y))

    return pos_y_process
```

Arquivo: database.py

```
import functions

import psutil
import platform
import os
import socket
import cpuinfo

### Mapping Function ###
def subnetwork_mapping(operational_system_data, subnetwork):
    hosts = functions.hosts_scan(operational_system_data, subnetwork)
    IPs = functions.ips_scan(hosts)
    hosts_data.append(hosts)
    ips_data.append(IPs)

### Network ###
ipv4s_data = list(functions.get_ip_addresses(socket.AF_INET))
ipv6s_data = list(functions.get_ip_addresses(socket.AF_INET6))
mac_address_data = list(functions.get_ip_addresses(psutil.AF_LINK))
subnetwork = ".".join(ipv4s_data[0][1].split('.')[0:3]) + '.'
hosts_data = list()
ips_data = list()
traffic_interface_data = functions.traffic_interface_scanner()

### PC ###
pc_name_data = platform.node()
operational_system_data = platform.system()
os_version_data = platform.version()
arc_data = platform.machine()

### CPU ###
cpu_data = cpuinfo.get_cpu_info()
p_cores_data = psutil.cpu_count(logical=False)
l_cores_data = psutil.cpu_count(logical=True)
cpu_percent_data = psutil.cpu_percent(interval=1)
cpu_cores_percent_data = psutil.cpu_percent(interval=1, percpu=True)
cpu_max_freq_data =  psutil.cpu_freq().max

### Memory ###
mem_data = psutil.virtual_memory()

### HD ###
pc_partitions_data = psutil.disk_partitions(all)
primary_storage_data = pc_partitions_data[0][0]
hd_data = psutil.disk_usage(str(primary_storage_data))

### Files Page ###
local_data = os.path.dirname(os.path.realpath(__file__))

## Process
processes_data = functions.scann_process_data()
```

Arquivo: server.py

```
import socket, psutil, pickle, platform, cpuinfo, os, nmap, threading, subprocess, datetime

### Functions ###
def subnetwork_mapping(operational_system_data, subnetwork):
    hosts = hosts_scan(operational_system_data, subnetwork)
    IPs = ips_scan(hosts)
    hosts_data.append(hosts)
    ips_data.append(IPs)

def get_ip_addresses(family):
    for interface, snics in psutil.net_if_addrs().items():
        for snic in snics:
            if snic.family == family:
                yield (interface, snic.address)

def ping(operational_system_data, hostname):
    args = []
    if operational_system_data == "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 hosts_scan(operational_system_data, subnetwork):
    print("Mapping", subnetwork, "subnetwork...\r")
    hosts_list = []
    return_codes = dict()
    for i in range(1, 255):
        host_test = (subnetwork + '{0}'.format(i))
        return_codes[subnetwork + '{0}'.format(i)] = ping(operational_system_data, host_test)
        if i %20 == 0:
            print(".", end = "")
        if return_codes[subnetwork + '{0}'.format(i)] == 0:
            hosts_list.append(subnetwork + '{0}'.format(i))
    print("\nMapping ready.")
    print("Hosts found:", hosts_list)
    return hosts_list

def ips_scan(hosts_list):
    nm = nmap.PortScanner()
    IPs = dict()
    for i in hosts_list:
        try:
            print("Scanning", i, "IP...")
            nm.scan(i)
            IPs[i] = nm[i]
            print("IP", i,"Scanned.")
        except:
            print("IP", i,"Failed.")
            continue
    print("Scanning Ready.")
    return IPs

def hosts_info(hosts_data, ips_data):
    IPs = dict()
    IPs_Info = ips_data[0]
    hosts = hosts_data[0]
    for i in hosts:
```

```
        IPs[i] = dict()
        try:
            IPs[i]['mac'] = IPs_Info[i]['addresses']['mac']
        except:
            pass
        try:
            IPs[i]['vendor'] = IPs_Info[i]['vendor'][IPs_Info[i]['addresses']['mac']]
        except:
            pass
        try:
            IPs[i]['response'] = IPs_Info[i]['status']['reason']
        except:
            pass
        try:
            IPs[i]['product'] = IPs_Info[i]['product']
            print(IPs[i]['product'])
        except:
            pass
    return IPs

def traffic_interface_scanner():
    io_status = psutil.net_io_counters(pernic=True)
    interface_names = list()
    for i in io_status:
        interface_names.append(str(i))
    interfaces_traffic = dict()
    for j in interface_names:
        interfaces_traffic[j] = dict()
        interfaces_traffic[j]['bytes_sent'] = io_status[j][0]
        interfaces_traffic[j]['bytes_recv'] = io_status[j][1]
        interfaces_traffic[j]['packets_sent'] = io_status[j][2]
        interfaces_traffic[j]['packets_recv'] = io_status[j][3]
    return interfaces_traffic

def scan_files(local_data):
    lista = os.listdir(local_data)
    files = dict()
    for i in lista:
        filepath = os.path.join(local_data, i)
        if os.path.isfile(filepath):
            files[i] = dict()
            files[i]['Size'] = os.stat(filepath).st_size
            files[i]['Created'] =
datetime.datetime.fromtimestamp(os.stat(filepath).st_ctime).strftime("%d-%m-%Y %H:%M:%S")
            files[i]['Mod'] =
datetime.datetime.fromtimestamp(os.stat(filepath).st_mtime).strftime("%d-%m-%Y %H:%M:%S")
    return files

def scann_process_data():
    processes = psutil.pids()
    process_data = dict()
    for i in processes:
        try:
            p = psutil.Process(i)
            proc_name = str(p.name())
            if proc_name not in process_data:
                process_data[proc_name] = dict()
                process_data[proc_name]['Status'] = list()
                process_data[proc_name]['PIDs'] = list()
                process_data[proc_name]['Create Time'] = list()
                process_data[proc_name]['Connections'] = list()
```

```
            process_data[proc_name]['Status'].append(p.status())
            process_data[proc_name]['PIDs'].append(i)
            process_data[proc_name]['Create
Time'].append(datetime.datetime.fromtimestamp(p.create_time()).strftime("%d-%m-%Y %H:%M:%S"))
            if p.connections() != []:
                process_data[proc_name]['Connections'].append(p.connections())
            else:
                process_data[proc_name]['Connections'].append('OFFLINE')
        except (psutil.AccessDenied, psutil.ZombieProcess, psutil.NoSuchProcess):
            continue

    all_process_data = dict()
    for i in process_data:
        all_process_data[i] = dict()
        all_process_data[i]['Qty PIDs'] = len(process_data[i]['PIDs'])
        all_process_data[i]['PIDs List'] = process_data[i]['PIDs']
        for j in range(all_process_data[i]['Qty PIDs']):
            all_process_data[i][process_data[i]['PIDs'][j]] = dict()
            all_process_data[i][process_data[i]['PIDs'][j]]['#'] = process_data[i]['PIDs'][j]
            all_process_data[i][process_data[i]['PIDs'][j]]['Status'] = process_data[i]['Status'][j]
            all_process_data[i][process_data[i]['PIDs'][j]]['Create Time'] = process_data[i]['Create
Time'][j]
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections'] = dict()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Connections Status'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Address Family'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Packets Type'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['LIP'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['LPort'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['RIP'] = list()
            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['RPort'] = list()
            if process_data[i]['Connections'][j] != 'OFFLINE':
                for c in process_data[i]['Connections'][j]:
                    all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Connections
Status'].append(c.status)
                    if c.family == socket.AF_INET:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Address
Family'].append('IPv4')
                    elif c.family == socket.AF_INET6:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Address
Family'].append('IPv6')
                    elif c.family == socket.AF_UNIX:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Address
Family'].append('Unix')
                    else:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Address
Family'].append('-')

                    if c.type == socket.SOCK_STREAM:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Packets
Type'].append('TCP')
                    elif c.type == socket.SOCK_DGRAM:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Packets
Type'].append('UDP')
                    elif c.type == socket.SOCK_RAW:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Packets
Type'].append('IP')
                    else:
                        all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Packets
Type'].append('-')

                    d_count = 1
```

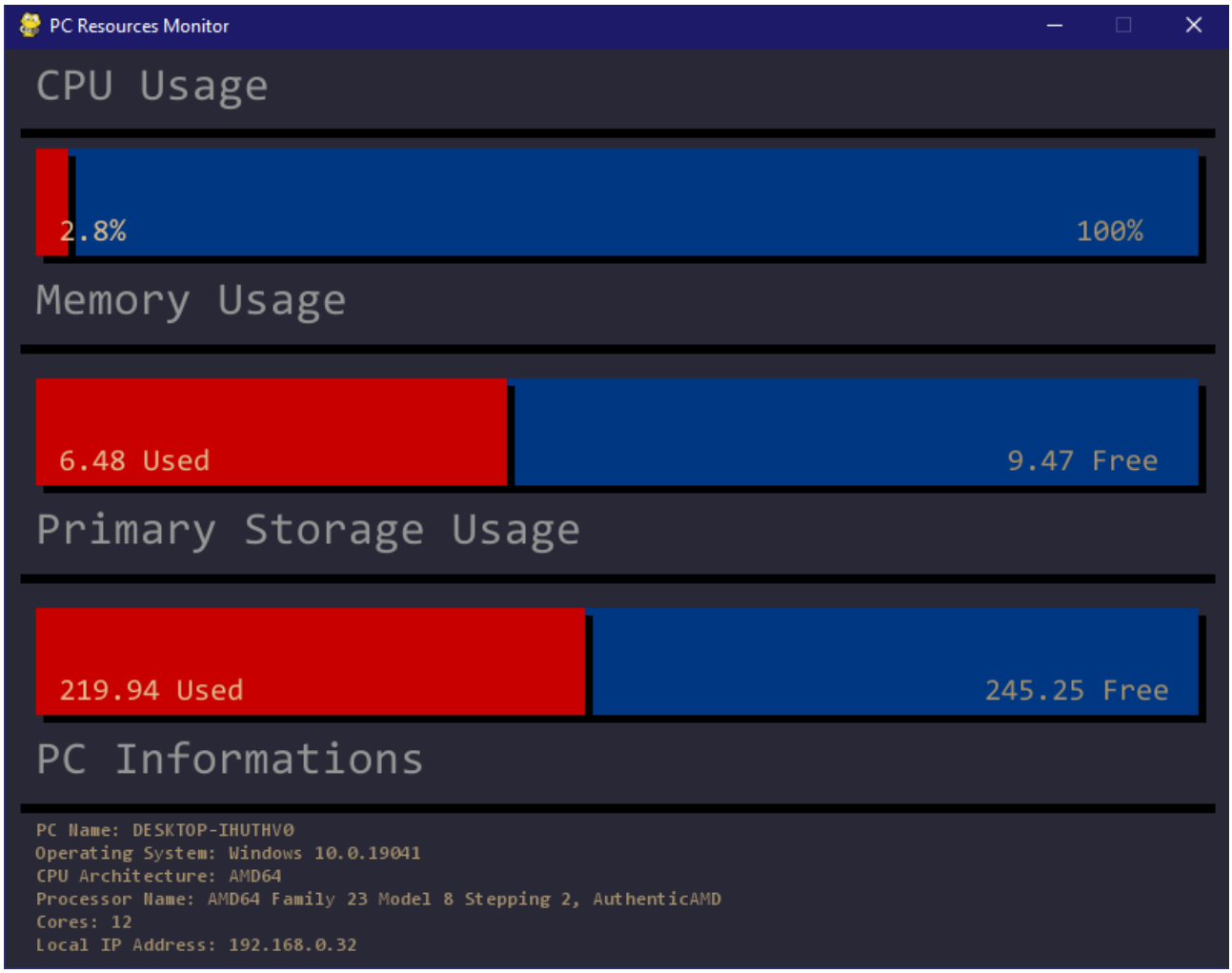
```
                    for d in c.laddr:
                        if d_count == 1:
                            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['LIP'].append(d)
                            d_count += 1
                        elif d_count == 2:

all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['LPort'].append(d)
                            d_count = 1
                    for d in c.raddr:
                        if d_count == 1:
                            all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['RIP'].append(d)
                            d_count += 1
                        elif d_count == 2:

all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['RPort'].append(d)
                            d_count = 1
            else:
                all_process_data[i][process_data[i]['PIDs'][j]]['Connections']['Connections Status'] =
process_data[i]['Connections'][j]
    return all_process_data

### DATA ###
# Network
ipv4s_data = list(get_ip_addresses(socket.AF_INET))
ipv6s_data = list(get_ip_addresses(socket.AF_INET6))
mac_address_data = list(get_ip_addresses(psutil.AF_LINK))
subnetwork = ".".join(ipv4s_data[0][1].split('.')[0:3]) + '.'
hosts_data = list()
ips_data = list()
traffic_interface_data = traffic_interface_scanner()
# PC
pc_name_data = platform.node()
operational_system_data = platform.system()
os_version_data = platform.version()
arc_data = platform.machine()
# CPU
cpu_info_data = cpuinfo.get_cpu_info()
p_cores_data = psutil.cpu_count(logical=False)
l_cores_data = psutil.cpu_count(logical=True)
# HD
pc_partitions_data = psutil.disk_partitions(all)
primary_storage_data = pc_partitions_data[0][0]
# Files Page
local_data = os.path.dirname(os.path.realpath(__file__))
## Process
processes_data = scann_process_data()

### Server ###
socket_server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()
port = 666
socket_server.bind((host, port))

socket_server.listen()
print("Server", host, "ready for connection on", port,"port.")
(socket_client,addr) = socket_server.accept()
print("Connected to:", str(addr))

hosts_mapped = False
while True:
    try:
```

```
        if not hosts_mapped:
            thread_mapping = threading.Thread(target=subnetwork_mapping, args=[operational_system_data,
subnetwork])
            thread_mapping.start()
            hosts_mapped = True
        pag_data = socket_client.recv(51200)
        data = dict()
        if pag_data.decode('utf-8') == 'HOME':
            data['cpu_percent'] = psutil.cpu_percent(interval=1)
            data['cpu_max_freq'] = psutil.cpu_freq().max
            data['mem'] = psutil.virtual_memory()
            data['hd'] = psutil.disk_usage(str(primary_storage_data))
            data['pc_name'] = pc_name_data
            data['os'] = operational_system_data
            data['os_version'] = os_version_data
            data['cpu_info'] = cpu_info_data
            data['cpu_arc'] = arc_data
            data['cpu_pcores'] = p_cores_data
            data['cpu_lcores'] = l_cores_data
        elif pag_data.decode('utf-8') == 'CPU':
            data['cpu_percent'] = psutil.cpu_percent(interval=1)
            data['cpu_max_freq'] = psutil.cpu_freq().max
            data['cpu_cores_percent'] = psutil.cpu_percent(interval=1, percpu=True)
            data['pc_name'] = pc_name_data
            data['os'] = operational_system_data
            data['os_version'] = os_version_data
            data['cpu_info'] = cpu_info_data
            data['cpu_arc'] = arc_data
            data['cpu_pcores'] = p_cores_data
            data['cpu_lcores'] = l_cores_data
        elif pag_data.decode('utf-8') == 'FILES':
            files_data = scan_files(local_data)
            data['files'] = files_data
        elif pag_data.decode('utf-8') == 'PROCESS':
            data['hosts'] = hosts_data
            data['ips'] = ips_data
            data['processes'] = processes_data
            data['traffic_interface'] = traffic_interface_scanner()
            data['ipv4s'] = ipv4s_data
            data['mac_address'] = mac_address_data
        bytes_resp = pickle.dumps(data)
        socket_client.send(bytes_resp)
    except:
        os._exit(0)
        socket_client.close()
        socket_server.close()
```

Anexo II

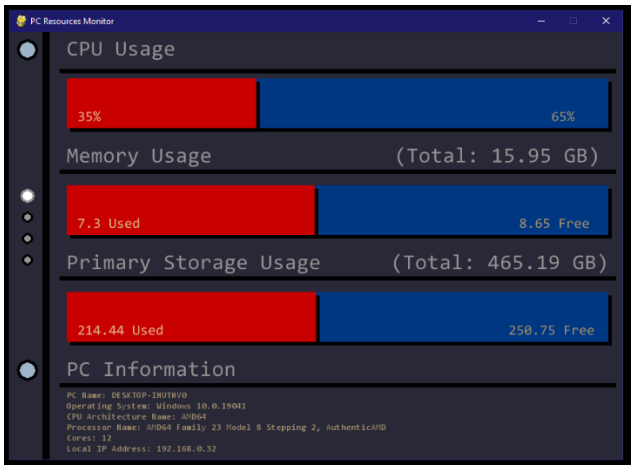
Impressões de Telas

Tela da Versão 1.0

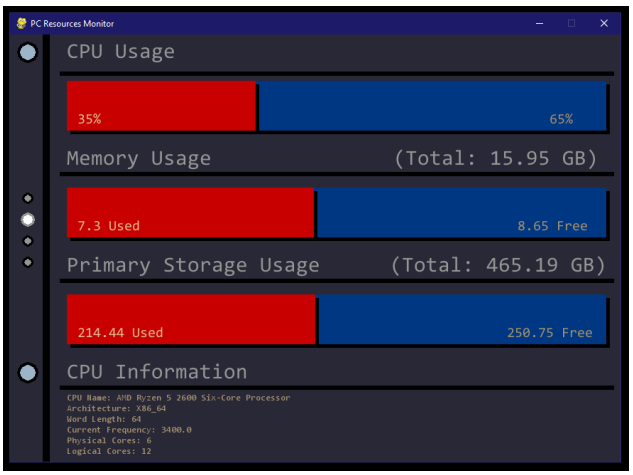


Tela do Software

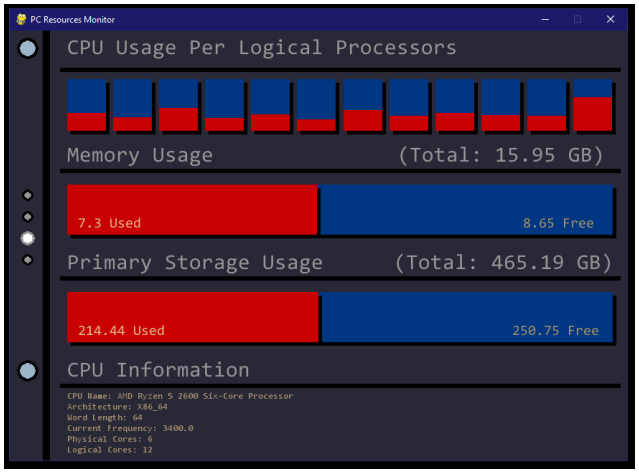
Telas da Versão 2.0



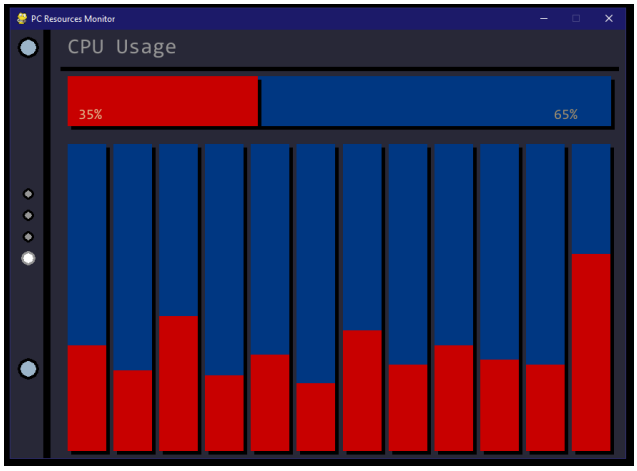
Tela Inicial



Segunda Tela

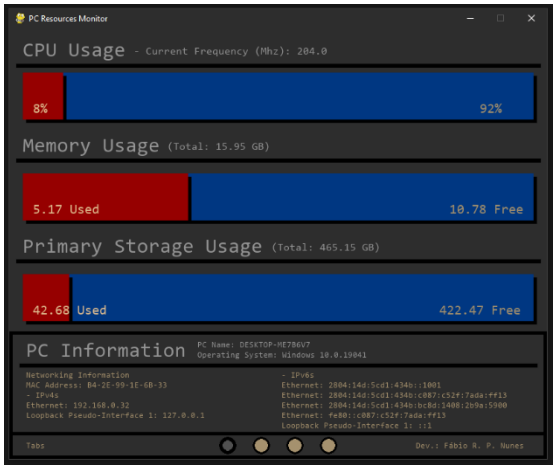


Quarta Tela

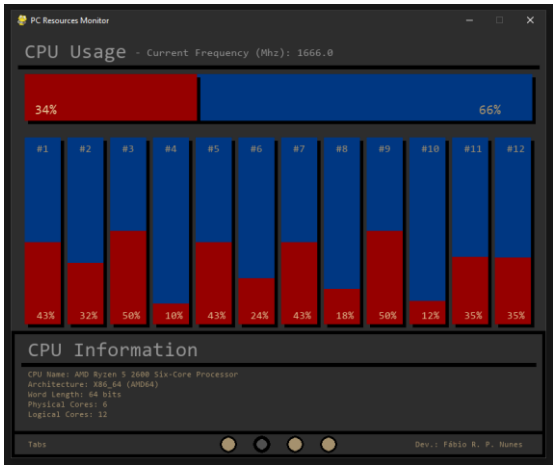


Quinta Tela

Telas da Versão 3.0

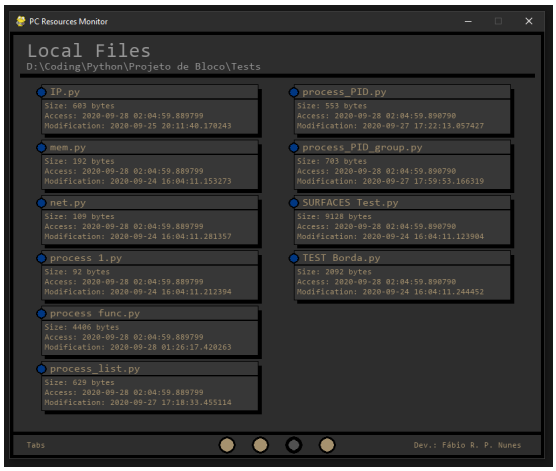


Tela de Informações Gerais

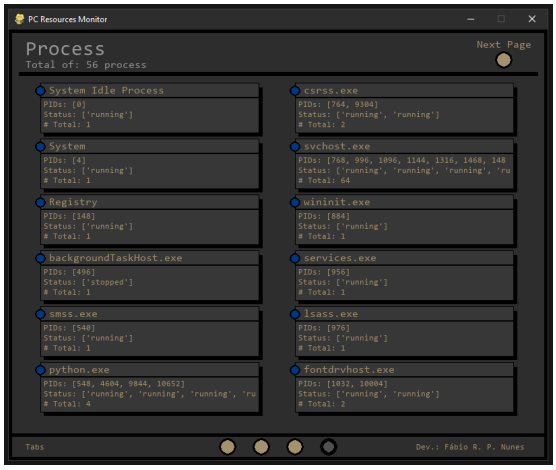


Tela de Informações da CPU

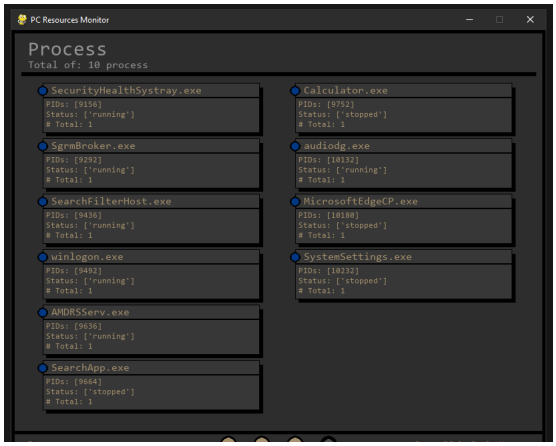
Tela de Arquivos, exibindo botão de página.



Tela de Arquivos, não exibindo botão de página.

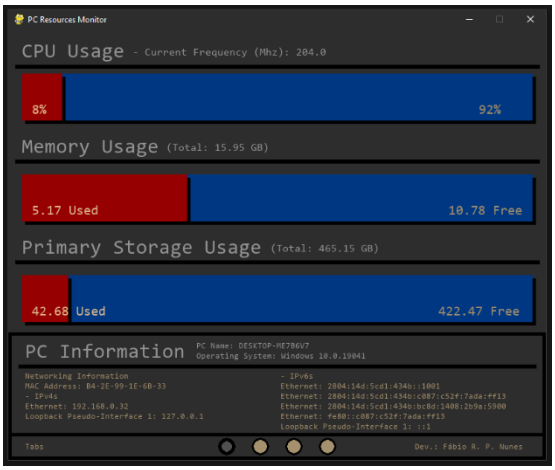


Tela de Processos, exibindo botão de página.

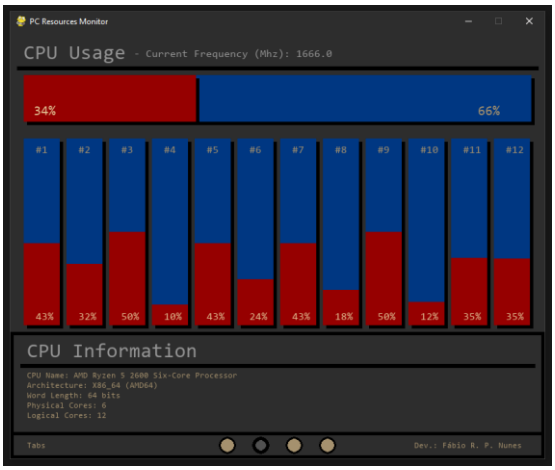


Tela de Processos, não exibindo botão de página.

Telas da Versão 4.0

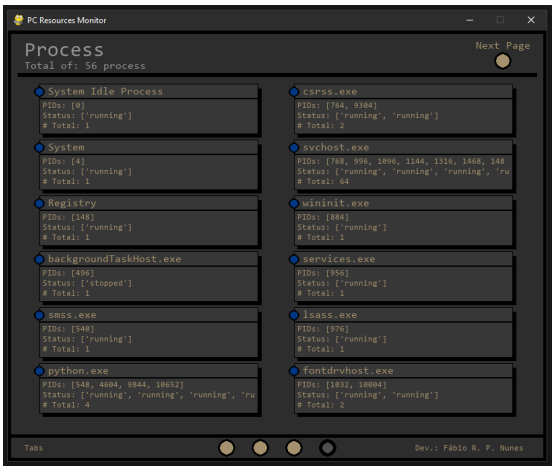


Tela de Informações Gerais



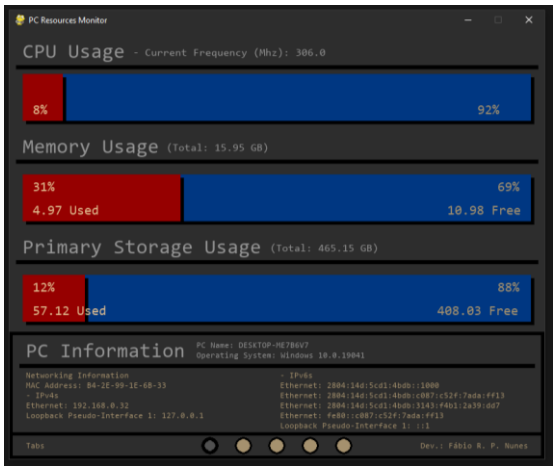
Tela de Informações de CPU

Tela de Arquivos

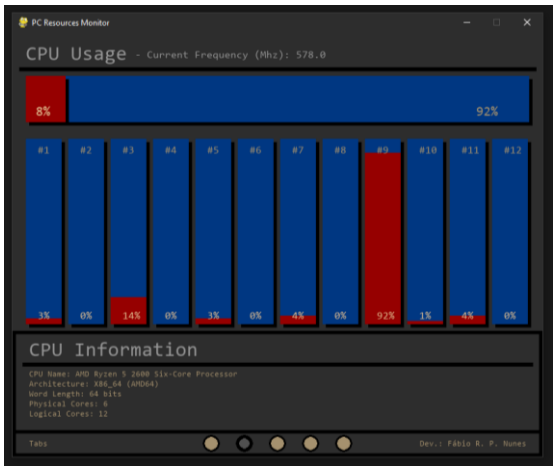


Tela de Processos

Telas da Versão 5.0

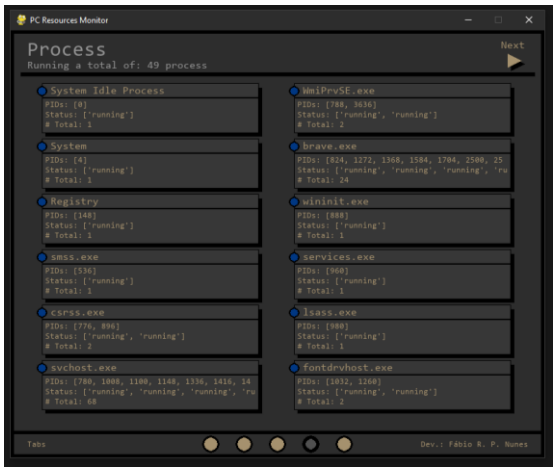


Tela de Informações Gerais



Tela de Informações da CPU

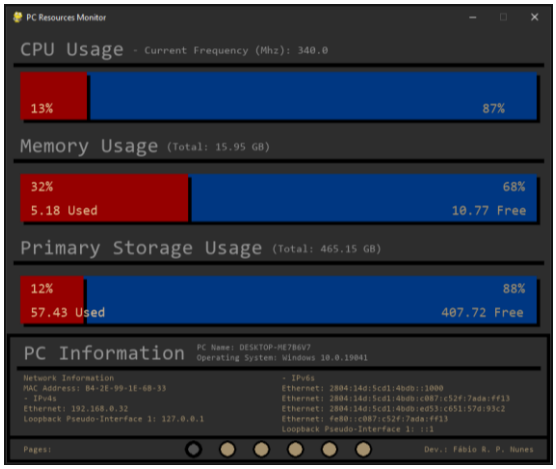
Tela de Arquivos



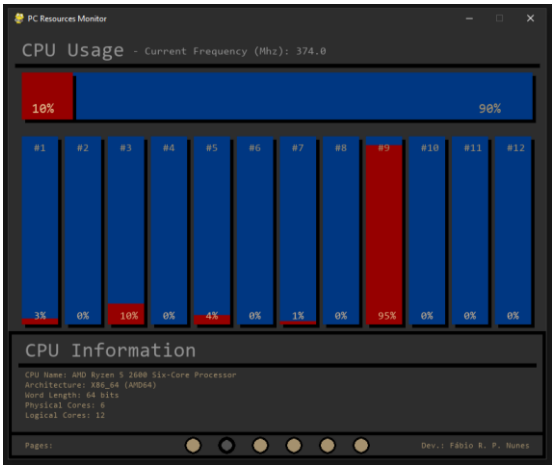
Tela de Processos

Tela de Informações de Rede

Telas da Versão 6.0



Tela de Informações Gerais

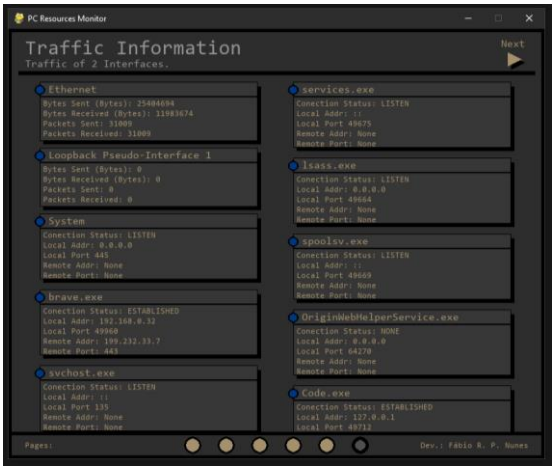


Tela de Informações da CPU

Tela de Arquivos

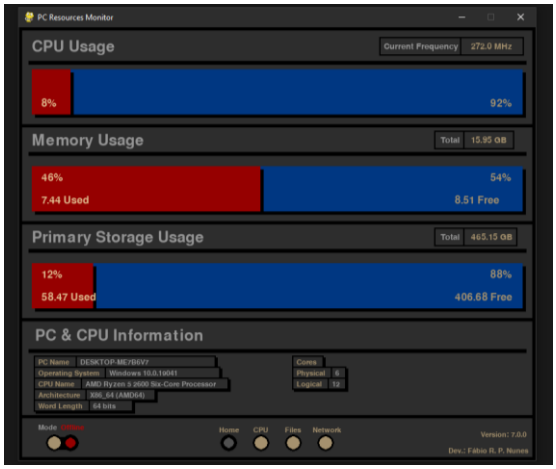
Tela de Processos

Tela de Informações de Rede

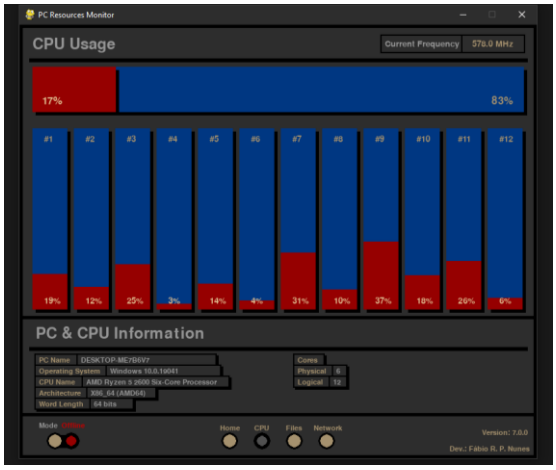


Tela de Informações de Tráfego

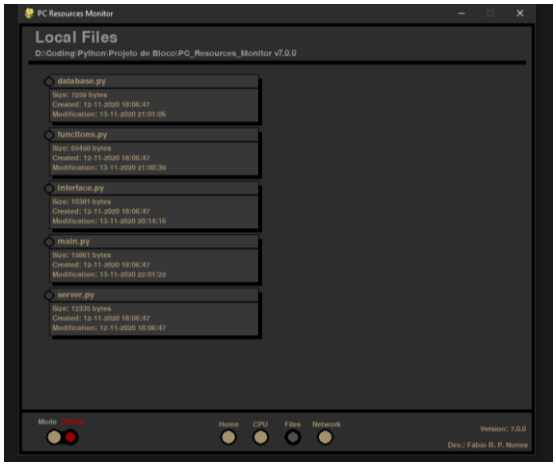
Telas da Versão 7.0



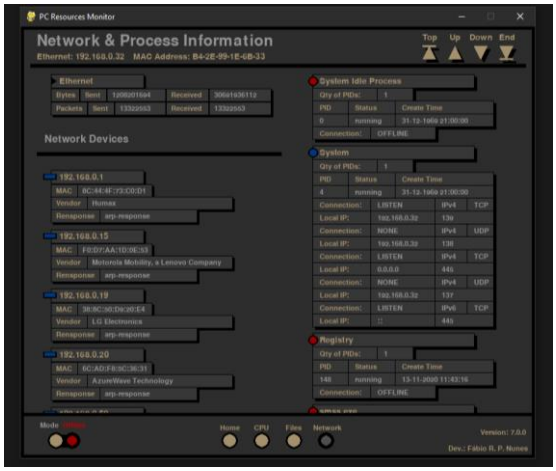
Tela de Informações Gerais



Tela de Informações da CPU



Tela de Arquivos



Tela de Rede e Processos