ADS - ANALISE E DESENVOLVIMENTO DE SISTEMAS

ARQUITETURA DE COMPUTADORES, REDES E SISTEMAS OPERACIONAIS

Projeto de Bloco Arquitetura de Computadores,
Sistemas Operacionais e Redes

Projeto de Bloco

Vinícius Mansoldo Walsh Ferreira

Rio de Janeiro,

02 de outubro de 2018

Introdução

A Walsh Enterprise apresenta seu mais novo projeto, Easy System, com ele o usuário poderá acessar de forma mais prática as informações de seu computador como memória, espaço no HD, seus arquivos e diretórios.

O sistema consiste em uma plataforma simplificada para pessoas com pouco conhecimento sobre o assunto, de forma intuitiva e com um simples menu, o usuário terá acesso as informações de seu sistema de forma prática e eficiente.

Desenvolvimento do Sistema

Projetado utilizando a tecnologia python 3.6, foram utilizadas algumas bibliotecas para auxiliar o desenvolvimento como: socket, pickle, cpuinfo, psutil e OS. O projeto fui feito em cima da comunicação via socket sendo o mesmo dividido em parte cliente e parte servidor.

Na parte cliente o usuário se depara com um menu simples e intuitivo onde basta alguns cliques e ele estará com as informações que precisa.

O trabalho ocorre no ‘back-end’ ou seja, na parte servidor onde estará sendo processadas as informações pedidas pelo usuário, utilizando as bibliotecas cpuinfo, psutil e OS, conseguimos pegar informações importantes da máquina do usuário e utilizando a biblioteca pickle enviamos os pacotes de dados (informações) do nosso servidor para nosso cliente.

Utilizamos a conexão TCP, pois queremos uma conexão segura entre de nosso sistema onde teremos a confiança de que toda informação pedida no lado cliente será recebida por nosso servidor e toda mensagem enviada pelo servidor chegará no cliente de maneira confiável e segura.

Como mencionado, o sistema e dividido em duas partes para maior otimização do mesmo, cliente e servidor, trabalhando de forma integrada para melhor servir o usuário.

Lado Cliente:

No lado cliente teremos funções que facilitam a formatação dos dados vindo do lado servidor de maneira que o usuário tenha todas as informações necessárias de uma forma prática e rápida. Sendo elas:

```
def formatar_processos_2():
    titulo = '{:^7}'.format("PID")
    titulo = titulo + '{:^11}'.format("# Threads")
    titulo = titulo + '{:^26}'.format("Criação")
    titulo = titulo + '{:^9}'.format("T. Usu.")
    titulo = titulo + '{:^9}'.format("T. Sis.")
    titulo = titulo + '{:^12}'.format("Mem. (%)")
    titulo = titulo + '{:^12}'.format("RSS")
    titulo = titulo + '{:^12}'.format("VMS")
    titulo = titulo + " Executável"
    print(titulo)
```

Esta função tem como objetivo, receber as informações do servidor e formata-las para que o usuário possa ver as informações dos processos ativos na máquina.

```
def formatar_redes(l):
    titulo = '{:11}'.format("Ip")
    titulo = titulo + '{:27}'.format("Netmask")
    titulo = titulo + '{:27}'.format("MAC")

    print(titulo)
```

Esta função tem como objetivo formatar as informações de rede vindas do servidor para o cliente.

Lado Servidor:

Do lado servidor teremos:

```
info = ("\n ====================MENU======================="
        "\n 1 - Informações da Máquina "
        "\n 2 - Informações de Arquivos "
        "\n 3 - Informações Processos Ativos "
        "\n 4 - Informações de Redes "
        "\n 5 - Sair "
        "\n ==============================================")
```

Nosso menu interativo com as funções para o usuário poder selecionar o que e necessário.

```
def encerrar_conexao(socket_cliente):
    info = ('Conexão Encerrada!')
    socket_cliente.send(info.encode('urf-8'))
    print("Fechando Conexão com", str(addr), "...")
    socket_cliente.shutdown(socket.SHUT_RDWR)
    socket_cliente.close()
```

Função que encerra a conexão com o cliente.

```
def info_redes(socket_cliente):
    dic_interfaces = psutil.net_if_addrs()
    ip = socket.gethostbyname(socket.gethostname())
    bytes_rep = pickle.dumps(dic_interfaces) #dicionario
    socket_cliente.send(bytes_rep)
```

Função que irá captar as informações de rede como IP /Endereço MAC e a máscara de rede.

```
def processos_ativos(socket_cliente):
    lista = psutil.pids() #pids
    bytes_rep = pickle.dumps(lista)
    socket_cliente.send(bytes_rep)  # Envia mensagem
```

Função que retorna uma lista com os processos ativos.

```
def arquivos_diretorios(socket_cliente):
    info2 = 'Usuario solicitou Informações sobre processos ativos'
    #obtém lista de arquivos e diretórios
    lista = os.listdir()#obtem lista arquivos diretorio
    #Cria um dicionário
    dic = {}
    for i in lista: #varia na lista dos arquivos e diretórios
        if os.path.isfile(i):#checa se é um arquivo
            dic[i] = []
            dic[i].append(os.stat(i).st_size)#tamanho
            dic[i].append(os.stat(i).st_atime)#tempo de criação
            dic[i].append(os.stat(i).st_mtime)#Tempo de Modificação
    bytes_rep = pickle.dumps(dic)
    # Envia os dados
    socket_cliente.send(bytes_rep)  # Envia mensagem
    print(info2)
```

Função que retorna o que existe no diretório, retornando as informações de tamanho, tempo de criação e ultima modificação.

```
 def info_disk(socket_cliente):
    disco = psutil.disk_usage('.')
    envia_dados = pickle.dumps(disco)
    socket_cliente.send(envia_dados)
```

Função que retorna as informações do disco do usuário.

```
def info_processador(socket_cliente):
    cpu_porc = psutil.cpu_count() #% de cpu por núcleos
    cpu_freq =psutil.cpu_freq().current #frequencia total
    cpu_nucleos = psutil.cpu_count(logical=False) #n° de núcleos e Threads
    resposta = {
        "cpu_logico": cpu_porc,
        "cpu_frequencia": cpu_freq,
        "cpu_fisico": cpu_nucleos
    }
    socket_cliente.send(pickle.dumps(resposta))
```

Função que retorna as informações do processador,

Estrutura do código

O sistema foi estruturado para ter a melhor leitura e facilidade de entendimento do código. nele temos a criação do socket inicialmente onde definimos nossa conexão entre cliente e servidor pela conexão TCP.

Escolhemos essa conexão pelo fato de garantir o envio e o recebimento dos pacotes de dados que serão enviados pelo cliente e pelo servidor, tendo assim segurança na conexão e no tráfego de informações de nosso sistema.

Após escolhermos a conexão TPC e a criarmos nosso socket, definimos a porta de conexão entre nosso servidor e o cliente, neste projeto usaremos a porta 6666.

Server:

```
# Cria o socket
socket_servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Obtem o nome da máquina
host = socket.gethostname()

porta = 6666

# Associa a porta
socket_servidor.bind((host, porta))

# Escutando
socket_servidor.listen(1)

print("Servidor:", host, "esperando conexão na porta", porta)

# Aceita alguma conexão
(cliente, addresse) = socket_servidor.accept()

print("Conectado a:", str(addresse))
```

Como podemos verificar, no lado servidor, teremos a criação do socket, a porta configurada, a associação entre a máquina host, ou seja, a máquina onde estará funcionando o servidor. e neste ponto, o servidor fica esperando uma conexão ser iniciada pelo cliente.

Após feita a conexão, o servidor avisa que está conectado ao endereço IP do cliente.

Basicamente iniciamos o servidor na mesma máquina ou em outra máquina e após o servidor estar ativo e esperando uma conexão iniciamos o cliente, esperamos a conexão ser feita e começamos a usufruir das informações.

Código Servidor:

```
import socket, psutil, pickle, os, cpuinfo

def uso_cpu_ram(socket_cliente):
    info1 =('Usuario solicitou Informações do computador')
    # Gera a lista de resposta
    resposta = []#lista dados
    resposta.append(psutil.cpu_percent())
    mem = psutil.virtual_memory()
    mem_percent = mem.used/mem.total * 100
    resposta.append(mem_percent)
    #Armazena Informações da CPU
    # info_cpu = cpuinfo.get_cpu_info()
    # socket_cliente.send(info_cpu)
    # Prepara a lista para o envio
    bytes_resp = pickle.dumps(resposta)#biblioteca compacta os arquivos ==>
1024 a 1024 bis-> bytes
    # Envia os dados
    socket_cliente.send(bytes_resp) # Envia mensagem
    print(info1)

def info_cpu(socket_cliente): #arquitewtura
    info = cpuinfo.get_cpu_info()
    envia_dados = pickle.dumps(info) #compacta dados - dicio
    socket_cliente.send(envia_dados)

def info_processador(socket_cliente):
    cpu_porc = psutil.cpu_count() #% de cpu por núcleos
    cpu_freq =psutil.cpu_freq().current #frequencia total
    cpu_nucleos = psutil.cpu_count(logical=False) #nº de núcleos e Threads
    resposta = {
        "cpu_logico": cpu_porc,
        "cpu_frequencia": cpu_freq,
        "cpu_fisico": cpu_nucleos
    }
    socket_cliente.send(pickle.dumps(resposta))

def info_disk(socket_cliente):
    disco = psutil.disk_usage('.')
    envia_dados = pickle.dumps(disco)
    socket_cliente.send(envia_dados)


def arquivos_diretorios(socket_cliente):
    info2 = 'Usuario solicitou Informações sobre processos ativos'
```

```
    #obtém lista de arquivos e diretórios
    lista = os.listdir()#obtem lista arquivos diretorio
    #Cria um dicionário
    dic = {}
    for i in lista: #varia na lista dos arquivos e diretórios
        if os.path.isfile(i):#checa se é um arquivo
            dic[i] = []
            dic[i].append(os.stat(i).st_size)#tamanho
            dic[i].append(os.stat(i).st_atime)#tempo de criação
            dic[i].append(os.stat(i).st_mtime)#Tempo de Modificação
    bytes_rep = pickle.dumps(dic)
    # Envia os dados
    socket_cliente.send(bytes_rep)  # Envia mensagem
    print(info2)

def processos_ativos(socket_cliente):
    lista = psutil.pids() #pids
    bytes_rep = pickle.dumps(lista)
    socket_cliente.send(bytes_rep)  # Envia mensagem


def info_redes(socket_cliente):
    dic_interfaces = psutil.net_if_addrs()
    ip = socket.gethostbyname(socket.gethostname())
    bytes_rep = pickle.dumps(dic_interfaces) #dicionario
    socket_cliente.send(bytes_rep)


def encerrar_conexao(socket_cliente):
    info = ('Conexão Encerrada!')
    socket_cliente.send(info.encode('urf-8'))
    print("Fechando Conexão com", str(addr), "...")
    socket_cliente.shutdown(socket.SHUT_RDWR)
    socket_cliente.close()

# Cria o socket
socket_servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Obtem o nome da máquina
host = socket.gethostname()
porta = 6666
# Associa a porta
socket_servidor.bind((host, porta))
# Escutando...
socket_servidor.listen()
print("Servidor", host, "esperando conexão na porta", porta)
# Aceita alguma conexão
(socket_cliente,addr) = socket_servidor.accept()
print("Conectado a:", str(addr))

info = ("\n ====================MENU======================="
        "\n 1 - Informações da Máquina "
        "\n 2 - Informações de Arquivos "
        "\n 3 - Informações Processos Ativos "
        "\n 4 - Informações de Redes "
        "\n 5 - Sair "
        "\n ================================================")
socket_cliente.send(info.encode('utf-8')) # Envia resposta

while True:
    # Decodifica mensagem em UTF-8:
    msg = socket_cliente.recv(1024)
    if '1' == msg.decode('utf-8'):
        uso_cpu_ram(socket_cliente)
        info_cpu(socket_cliente)
        info_processador(socket_cliente)
        info_disk(socket_cliente)
```

```
    elif '2' == msg.decode('utf-8'):
        arquivos_diretorios(socket_cliente)
        print('O Usuário Solicitou Informações sobre Arquivos.')
    elif '3' == msg.decode('utf-8'):
        processos_ativos(socket_cliente)
        print('O usuário solicitou informações sobre processos.')
    elif '4' == msg.decode('utf-8'):
        info_redes(socket_cliente)
        print('O Usuário solicitou informações de redes')
    elif '5' == msg.decode('utf-8'):
        encerrar_conexao(socket_cliente)
        break
    else:
        print('O usuário Digitou opções inválidas.')
```

Código Cliente:

```
import socket, time, pickle, psutil

# Função que Imprime a lista formatada
def Imprimir(l):
    texto = ''
    for i in l:
        texto = texto + '{:>8.2f}'.format(i)
    print(texto)

def formatar_processos_2():
    titulo = '{:^7}'.format("PID")
    titulo = titulo + '{:^11}'.format("# Threads")
    titulo = titulo + '{:^26}'.format("Criação")
    titulo = titulo + '{:^9}'.format("T. Usu.")
    titulo = titulo + '{:^9}'.format("T. Sis.")
    titulo = titulo + '{:^12}'.format("Mem. (%)")
    titulo = titulo + '{:^12}'.format("RSS")
    titulo = titulo + '{:^12}'.format("VMS")
    titulo = titulo + " Executável"
    print(titulo)

def formatar_processos_1(pid):
    try:
        p = psutil.Process(pid)
        texto = '{:6}'.format(pid)
        texto = texto + '{:11}'.format(p.num_threads())
        texto = texto + " " + time.ctime(p.create_time()) + " "
        texto = texto + '{:8.2f}'.format(p.cpu_times().user)
        texto = texto + '{:8.2f}'.format(p.cpu_times().system)
        texto = texto + '{:10.2f}'.format(p.memory_percent()) + " MB"
        rss = p.memory_info().rss / 1024 / 1024
        texto = texto + '{:10.2f}'.format(rss) + " MB"
        vms = p.memory_info().vms / 1024 / 1024
        texto = texto + '{:10.2f}'.format(vms) + " MB"
        texto = texto + " " + p.exe()
        print(texto)
    except:
        pass


def formatar_redes(l):
    titulo = '{:11}'.format("Ip")
    titulo = titulo + '{:27}'.format("Netmask")
    titulo = titulo + '{:27}'.format("MAC")

    print(titulo)
```

```
# Cria o socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Tenta se conectar ao servidor
s.connect((socket.gethostname(), 6666))

controle = True

menu = s.recv(1024)

while controle:
    print(menu.decode('utf-8'))
    msg1 = input('digite a opção desejada: ')
    if msg1 == '1':
        s.send(msg1.encode('utf-8'))
        msg = ' '
        print('{:>8}'.format('%CPU')+'{:>8}'.format('%MEM'))
        # Envia mensagem vazia apenas para indicar a requisição

        bytes = s.recv(1024)
        # Converte os bytes para lista

        lista = pickle.loads(bytes)
        #Uso de Cpu e memoria em Porcentagem#
        Imprimir(lista)

        info_cpu = s.recv(1024)
        info = pickle.loads(info_cpu)

        # Informações do Processador#
        print('Processador:', info['brand'])  # brand
        print('Arquitetura:', info['arch'])  # arch
        print('Bits: ', info['bits'])  # bits
        cpu = s.recv(1024)
        cpu_logico = pickle.loads(cpu)["cpu_logico"]
        print('Núcleos Lógicos:', cpu_logico) # de cpu por núcleos
        cpu_frequencia = pickle.loads(cpu)["cpu_frequencia"]
        print('Frequência:', cpu_frequencia) #frequencia total
        cpu_fisico = pickle.loads(cpu)["cpu_fisico"]
        print('Núcleos Físicos:', cpu_fisico) #n° de núcleos e Threads
        recebe_disco = s.recv(1024)
        disco = pickle.loads(recebe_disco)
        print("Percentual de Disco Usado:", disco.percent, '%')

    elif msg1 == '2':
        s.send(msg1.encode('utf-8'))
        bytes = s.recv(2048)
        lista2 = pickle.loads(bytes)
        titulo = '{:11}'.format("Tamanho") # 10 caracteres + 1 de espaço
        # Concatenar com 25 caracteres + 2 de espaços
        titulo = titulo + '{:27}'.format("Data de Modificação")
        # Concatenar com 25 caracteres + 2 de espaços
        titulo = titulo + '{:27}'.format("Data de Criação")
        titulo = titulo + "Nome"
        print(titulo)
        for i in lista2:
            kb = lista2[i][0]/1000
            tamanho = '{:10}'.format(str('{:.2f}'.format(kb)+' KB'))
            print(tamanho, time.ctime(lista2[i][2]), " ",
time.ctime(lista2[i][1]), " ", i)

    elif msg1 == '3':
        s.send(msg1.encode('utf-8'))
        bytes = s.recv(1024)
        dic = pickle.loads(bytes)
        lista = psutil.pids()
```

```
        formatar_processos_2()
        for i in lista:
            formatar_processos_1(i)

    elif msg1 == '4':
        s.send(msg1.encode('utf-8'))
        bytes = s.recv(2048)
        dic_redes = pickle.loads(bytes)
        print('Endereço de Rede:',dic_redes['Ethernet'][1].address)
        print('Mascara de Rede:',dic_redes['Ethernet'][1].netmask)
        print('MAC:',dic_redes['Ethernet'][0].address)

    elif msg1 == '5':
        s.send(msg1.encode('utf-8'))
        bytes = s.recv(1024)
        s.shutdown(socket.SHUT_RDWR)
        s.close()
        controle = False
    else:
        print('Opção inválida!')
```

Resultado e análises

Vamos agora rodar o servidor e o cliente e ver como esta sendo apresentado as respostas:

Cliente:

```
====================MENU=======================
1 - Informações da Máquina
2 - Informações de Arquivos
3 - Informações Processos Ativos
4 - Informações de Redes
5 - Sair
================================================
digite a opção desejada:
```

Menu que será apresentado para o cliente. Lembrando de iniciar primeiro o servidor para evitar erro.

Servidor:

```
Servidor Walsh-PC esperando conexão na porta 6666
Conectado a: ('169.254.19.198', 4099)
```

Inicialmente o servidor estará esperando conexão na porta designada, após o cliente estiver conectado, ele irá avisar que o cliente de ip está conectado ao servidor.

Selecionando a primeira opção o cliente irá receber informações de sua máquina:

```
 ====================MENU=======================
 1 - Informações da Máquina
 2 - Informações de Arquivos
 3 - Informações Processos Ativos
 4 - Informações de Redes
 5 - Sair
 ================================================
digite a opção desejada: 1
    %CPU    %MEM
   11.10   56.68
Processador: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz
Arquitetura: X86_64
Bits:  64
Núcleos Lógicos: 4
Frequência: 2300.0
Núcleos Físicos: 2
Percentual de Disco Usado: 20.2 %
```

Selecionando a segunda opção o usuário terá as informações dos arquivos da pasta onde se encontra o servidor.

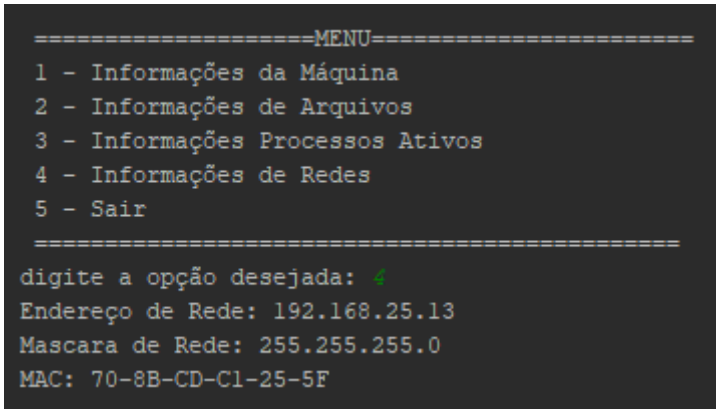
```
 ====================MENU=======================
 1 - Informações da Máquina
 2 - Informações de Arquivos
 3 - Informações Processos Ativos
 4 - Informações de Redes
 5 - Sair
 ================================================
digite a opção desejada: 2
Tamanho    Data de Modificação        Data de Criação            Nome
4.49 KB    Tue Oct  2 14:17:41 2018   Tue Oct  2 14:17:41 2018   Cliente.py
4.40 KB    Mon Oct  1 21:49:42 2018   Mon Oct  1 20:51:49 2018   Servidor.py
```

Selecionando a terceira opção o usuário terá acesso aos PIDs, processos que estão no momento ativos na máquina.

Exemplo da lista:

```
 ====================MENU=======================
 1 - Informações da Máquina
 2 - Informações de Arquivos
 3 - Informações Processos Ativos
 4 - Informações de Redes
 5 - Sair
 ================================================
digite a opção desejada: 3
  PID   # Threads          Criação          T. Usu.  T. Sis.   Mem. (%)      RSS        VMS      Executável
 4612         4 Mon Oct  1 22:16:18 2018     0.06    0.17      0.02 MB     1.34 MB     1.28 MB C:\Windows\Temp\DPTF\esif_assist_64.exe
  848        28 Mon Oct  1 22:16:19 2018     3.02    2.33      0.13 MB    10.73 MB    23.17 MB C:\Program Files\NVIDIA Corporation\NvContainer\nvcontainer.exe
 7160        11 Mon Oct  1 22:16:20 2018     3.91    4.44      0.22 MB    18.12 MB     6.99 MB C:\Windows\System32\sihost.exe
 2044         8 Mon Oct  1 22:16:25 2018     0.31    0.80      0.12 MB     9.88 MB     6.25 MB C:\Windows\System32\dllhost.exe
 4080        13 Mon Oct  1 22:16:28 2018     5.53    1.47      0.19 MB    15.50 MB     8.65 MB C:\Windows\System32\svchost.exe
 3412         8 Mon Oct  1 22:16:29 2018     5.80    1.81      0.26 MB    21.01 MB     7.21 MB C:\Windows\System32\svchost.exe
  636         6 Mon Oct  1 22:16:32 2018     0.30    0.88      0.07 MB     5.63 MB     2.98 MB C:\Windows\System32\taskhostw.exe
 7292        98 Mon Oct  1 22:16:39 2018    42.84   54.52      1.24 MB   100.02 MB    66.21 MB C:\Windows\explorer.exe
 7940         6 Mon Oct  1 22:17:03 2018     0.11    0.12      0.02 MB     1.70 MB     6.34 MB C:\Program Files\Realtek\Audio\HDA\RAVBg64.exe
 7984         9 Mon Oct  1 22:17:04 2018     0.16    0.50      0.20 MB    16.49 MB     4.45 MB C:\Program Files\Realtek\Audio\HDA\RAVCpl64.exe
 5532         4 Mon Oct  1 22:17:35 2018     0.09    0.36      0.06 MB     4.62 MB     3.49 MB C:\Windows\System32\DriverStore\FileRepository\igdlh64.inf_amd64_463164d40c3d26ce\igfxEM.exe
 3820        26 Mon Oct  1 22:18:24 2018     6.20    3.58      0.96 MB    77.52 MB    37.48 MB C:\Windows\SystemApps\ShellExperienceHost_cw5n1h2txyewy\ShellExperienceHost.exe
 7136        38 Mon Oct  1 22:18:28 2018    11.02    6.91      1.96 MB   158.52 MB    87.29 MB C:\Windows\SystemApps\Microsoft.Windows.Cortana_cw5n1h2txyewy\SearchUI.exe
 6636         6 Mon Oct  1 22:18:29 2018     1.66    1.09      0.25 MB    19.83 MB     7.40 MB C:\Windows\System32\RuntimeBroker.exe
 7044         4 Mon Oct  1 22:18:32 2018     0.17    0.25      0.13 MB    10.16 MB     6.23 MB C:\Windows\System32\svchost.exe
 3644        91 Mon Oct  1 22:18:35 2018     1.38    1.05      0.22 MB    17.48 MB    33.41 MB C:\Program Files (x86)\NVIDIA Corporation\NvNode\NVIDIA Web Helper.exe
 8232        30 Mon Oct  1 22:18:36 2018     2.59    2.55      0.23 MB    18.94 MB     9.16 MB C:\Windows\System32\RuntimeBroker.exe
 8412         3 Mon Oct  1 22:18:36 2018     0.05    0.08      0.00 MB     0.29 MB     5.34 MB C:\Windows\System32\conhost.exe
 8756        35 Mon Oct  1 22:18:40 2018     0.81    1.47      0.09 MB     7.62 MB    35.55 MB C:\Program Files\WindowsApps\Microsoft.SkypeApp_12.1815.210.0_x64__kzf8qxf38zg5c\SkypeHost.exe
 8976         6 Mon Oct  1 22:18:44 2018    11.47   10.89      0.03 MB     2.06 MB    15.30 MB C:\Windows\System32\SettingSyncHost.exe
```

Com a quarta opção o usuário tem acesso as informações de sua rede, seu endereço de ip, máscara de rede e endereço MAC.



Na ultima opção, o usuário fecha a conexão com o servidor, terminando assim com a transferência de informações.

Conclusão

Podemos concluir que o sistema e de fácil manuseio, de estrutura de fácil acesso, com boa leitura de código e boa manutenção para o mesmo.

Com uma conexão TCP, temos a certeza que teremos nossas informações seguras sendo passadas entre o cliente e o servidor.

Para uma versão alpha de desenvolvimento para usuários internos e pessoas com algum nível de conhecimento, podemos dizer que o projeto foi bem sucedido e com um grau de aproveitamento e com bons feedbacks para a próxima versão Beta, onde a ideia será começar a criar mais opções no menu, e futuramente um menu gráfico com funções mais interativas com o usuário final.