INSTITUTO INFNET

ENGENHARIA DE SOFTWARE – EDS-2017-N2

ALEXANDER VIEIRA DA SILVA

MATRÍCULA 041.380.007-55

PROJETO DE BLOCO - ARQUITETURA DE COMPUTADORES, SISTEMAS OPERACIONAIS E REDES

TESTE DE PERFORMANCE – TP6

RIO DE JANEIRO-RJ

2018

Este TP6 corresponde à continuação do TP5. Agora, você irá introduzir no seu programa informações sobre arquivos e diretórios especificados e sobre processos em execução no computador. A partir de então, você terá mais liberdade para criar a forma de visualização das informações da maneira que desejar e de acordo com seu orientador. Certifique-se apenas que está realizando o básico do que este TP requisita.

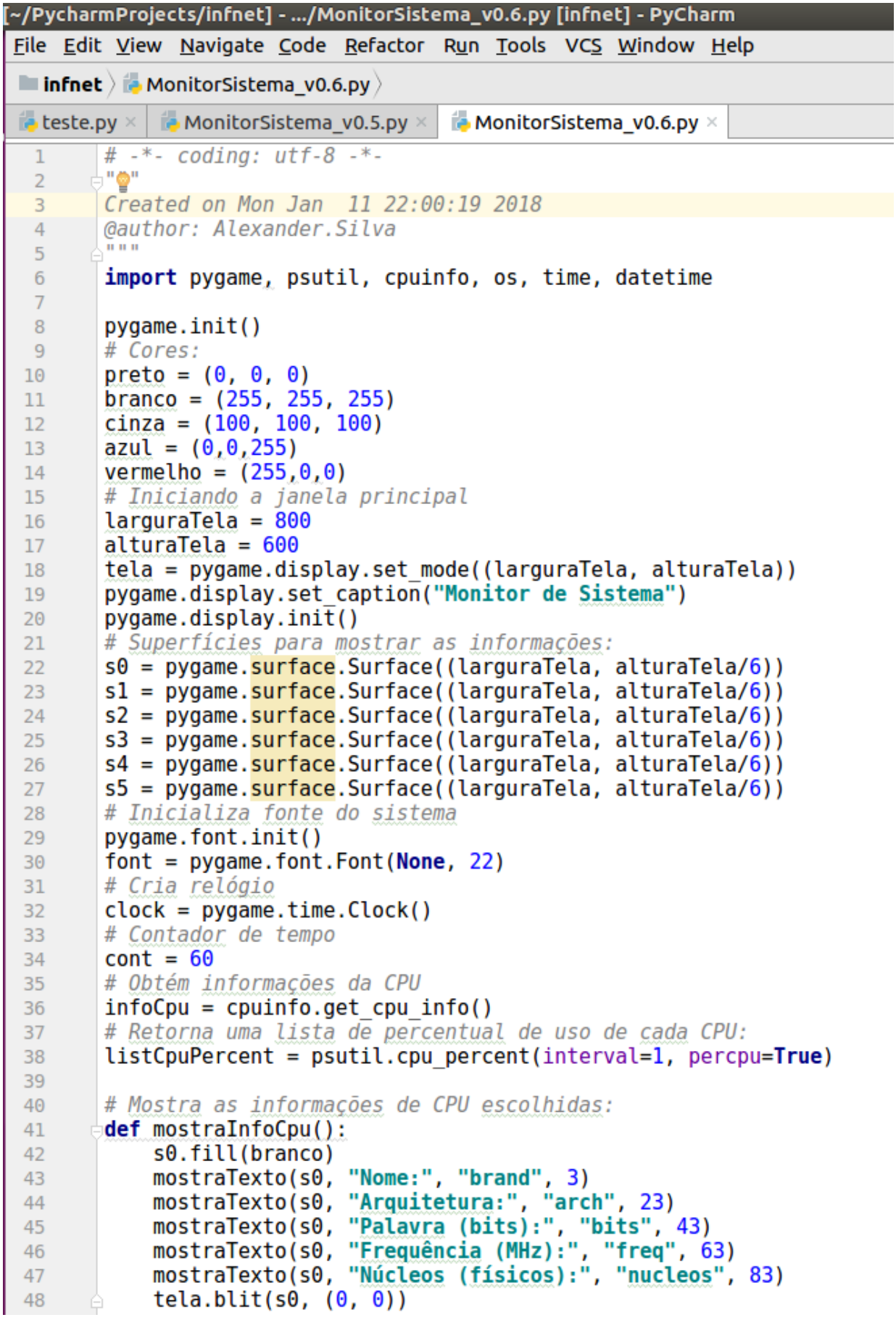
A seguir, os requisitos básicos:

1.Criar uma ou mais funções que retornem ou apresentem informações sobre diretórios e arquivos. Tais informações podem ser qualquer uma que você achar relevante disponível no módulo ‘os’ e ‘psutil’ de Python, como nome, tamanho, localização, data de criação, data de modificação, tipo, etc.

2.Usar a função em seu programa para mostrar o resultado. O resultado pode ser em texto formatado impresso na tela ou gráfico, usando o Pygame. Note que o uso do Pygame é opcional.

3.Criar uma ou mais funções que retornem ou apresentem informações sobre processos do sistema. As informações podem ser: PID, nome do executável, consumo de processamento, consumo de memória, entre outras disponíveis no módulo ‘psutil’ de Python.

4.Usar a função em seu programa para mostrar o resultado. O resultado pode ser em texto formatado impresso na tela ou gráfico, usando o Pygame. Note que o uso do Pygame é opcional.


[~/PycharmProjects/infnet] - .../MonitorSistema_v0.6.py [infnet] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
infnet
MonitorSistema_v0.6.py
teste.py
MonitorSistema_v0.5.py
MonitorSistema_v0.6.py
-*- coding: utf-8 -*-
"""
Created on Mon Jan 11 22:00:19 2018
@author: Alexander.Silva
"""
import pygame, psutil, cpuinfo, os, time, datetime
pygame.init()
# Cores:
preto = (0, 0, 0)
branco = (255, 255, 255)
cinza = (100, 100, 100)
azul = (0,0,255)
vermelho = (255,0,0)
# Iniciando a janela principal
larguraTela = 800
alturaTela = 600
tela = pygame.display.set_mode((larguraTela, alturaTela))
pygame.display.set_caption("Monitor de Sistema")
pygame.display.init()
# Superfícies para mostrar as informações:
s0 = pygame.surface.Surface((larguraTela, alturaTela/6))
s1 = pygame.surface.Surface((larguraTela, alturaTela/6))
s2 = pygame.surface.Surface((larguraTela, alturaTela/6))
s3 = pygame.surface.Surface((larguraTela, alturaTela/6))
s4 = pygame.surface.Surface((larguraTela, alturaTela/6))
s5 = pygame.surface.Surface((larguraTela, alturaTela/6))
# Inicializa fonte do sistema
pygame.font.init()
font = pygame.font.Font(None, 22)
# Cria relógio
clock = pygame.time.Clock()
# Contador de tempo
cont = 60
# Obtém informações da CPU
infoCpu = cpuinfo.get_cpu_info()
# Retorna uma lista de percentual de uso de cada CPU:
listCpuPercent = psutil.cpu_percent(interval=1, percpu=True)
# Mostra as informações de CPU escolhidas:
def mostraInfoCpu():
s0.fill(branco)
mostraTexto(s0, "Nome:", "brand", 3)
mostraTexto(s0, "Arquitetura:", "arch", 23)
mostraTexto(s0, "Palavra (bits):", "bits", 43)
mostraTexto(s0, "Frequência (MHz):", "freq", 63)
mostraTexto(s0, "Núcleos (físicos):", "nucleos", 83)
tela.blit(s0, (0, 0))

~/PycharmProjects/infnet] - .../MonitorSistema_v0.6.py [infnet] - PyCharm

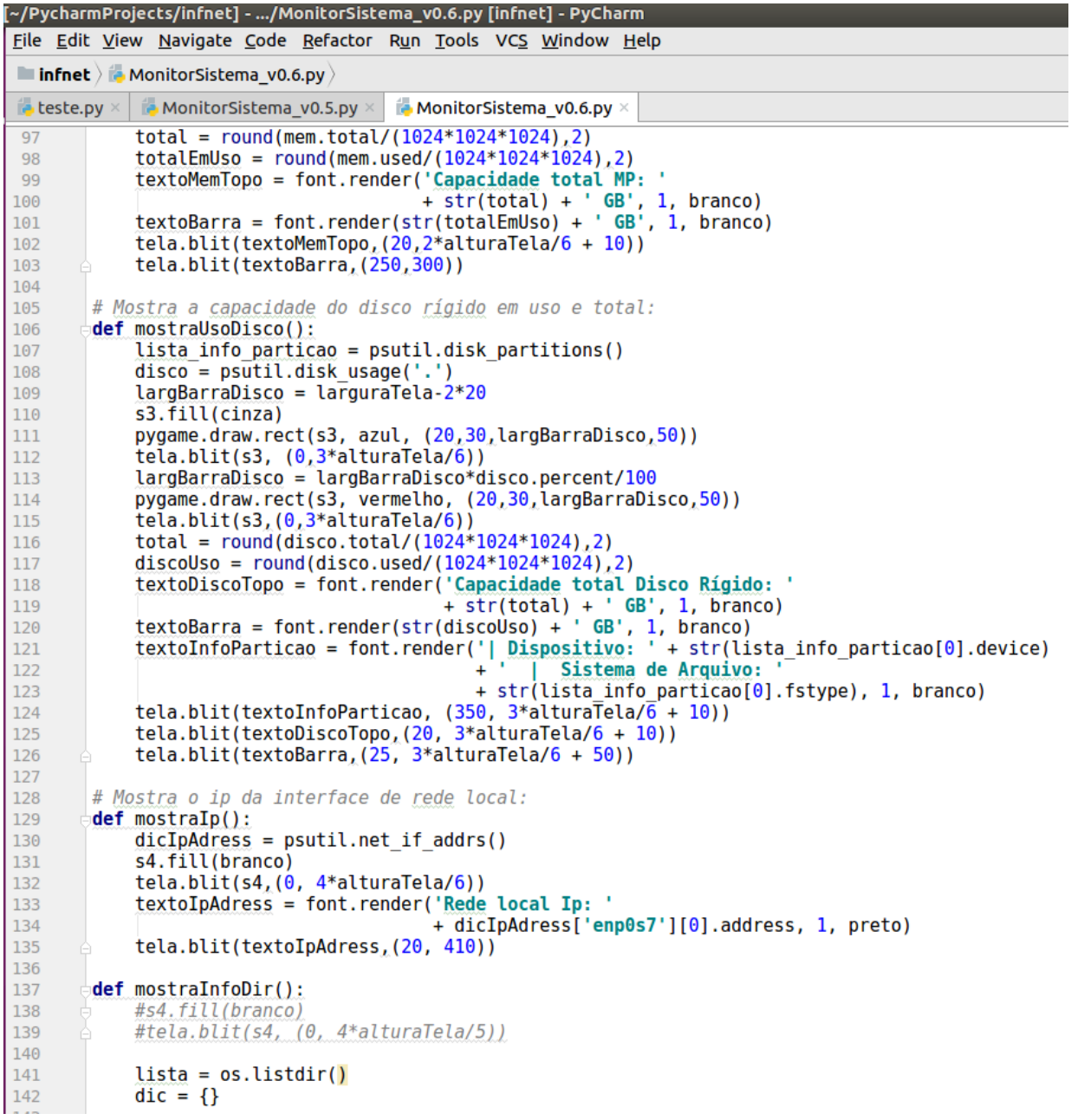
File Edit View Navigate Code Refactor Run Tools VCS Window Help

infnet > MonitorSistema_v0.6.py

teste.py | MonitorSistema_v0.5.py | MonitorSistema_v0.6.py

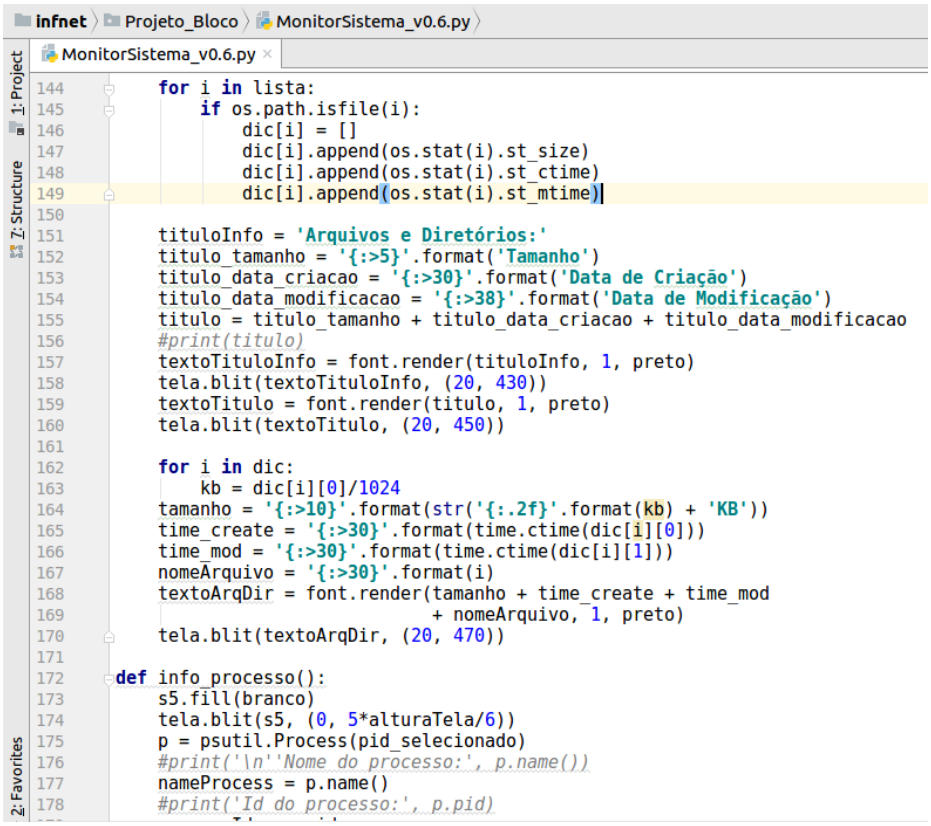
```
49
50  # Mostra texto de acordo com uma chave:
51  def mostraTexto(s0, nome, chave, pos_y):
52      text = font.render(nome, True, preto)
53      s0.blit(text, (10, pos_y))
54      if chave == "freq":
55          s1 = str(round(psutil.cpu_freq().current, 2))
56      elif chave == "nucleos":
57          s1 = str(psutil.cpu_count())
58          s1 = s1 + " (" + str(psutil.cpu_count(logical=True)) + ")"
59      else:
60          s1 = str(infoCpu[chave])
61      text = font.render(s1, True, cinza)
62      s0.blit(text, (160, pos_y))
63
64  # Mostra o percentual de uso de CPU:
65  def mostraUsoCpu(s1, listCpuPercent):
66      s1.fill(cinza)
67      numCpu = len(listCpuPercent)
68      x = y = 10
69      desl = 10
70      alt = s0.get_height()-2*y
71      larg = (s0.get_width()-2*x - (numCpu+1)*desl)/numCpu
72      d = x + desl
73      for i in listCpuPercent:
74          pygame.draw.rect(s0, vermelho, (d, y, larg, alt))
75          pygame.draw.rect(s0, azul, (d, y, larg, (1-i/100)*alt))
76          d = d + larg + desl
77          tela.blit(s0, (0, alturaTela/6))
78      textoCpu1 = font.render('Core 1: ' + str(listCpuPercent[0]) + ' %', 1, branco)
79      tela.blit(textoCpu1,(60, alturaTela/6 + 50))
80      textoCpu2 = font.render('Core 2: ' + str(listCpuPercent[1]) + ' %', 1, branco)
81      tela.blit(textoCpu2,(260, alturaTela/6 + 50))
82      textoCpu3 = font.render('Core 3: ' + str(listCpuPercent[2]) + ' %', 1, branco)
83      tela.blit(textoCpu3,(440, alturaTela/6 + 50))
84      textoCpu4 = font.render('Core 4: ' + str(listCpuPercent[3]) + ' %', 1, branco)
85      tela.blit(textoCpu4,(640, alturaTela/6 + 50))
86
87  # Mostra a capacidade de MP em uso e total:
88  def mostraUsoMemoria():
89      mem = psutil.virtual_memory()
90      largBarraMem = larguraTela-2*20
91      s2.fill(cinza)
92      pygame.draw.rect(s2, azul, (20,30,largBarraMem,50))
93      tela.blit(s2, (0,2*alturaTela/6))
94      largBarraMem = largBarraMem*mem.percent/100
95      pygame.draw.rect(s2, vermelho, (20,30,largBarraMem,50))
96      tela.blit(s2,(0,2*alturaTela/6))
```

Resposta para os requisitos 1 e 2:

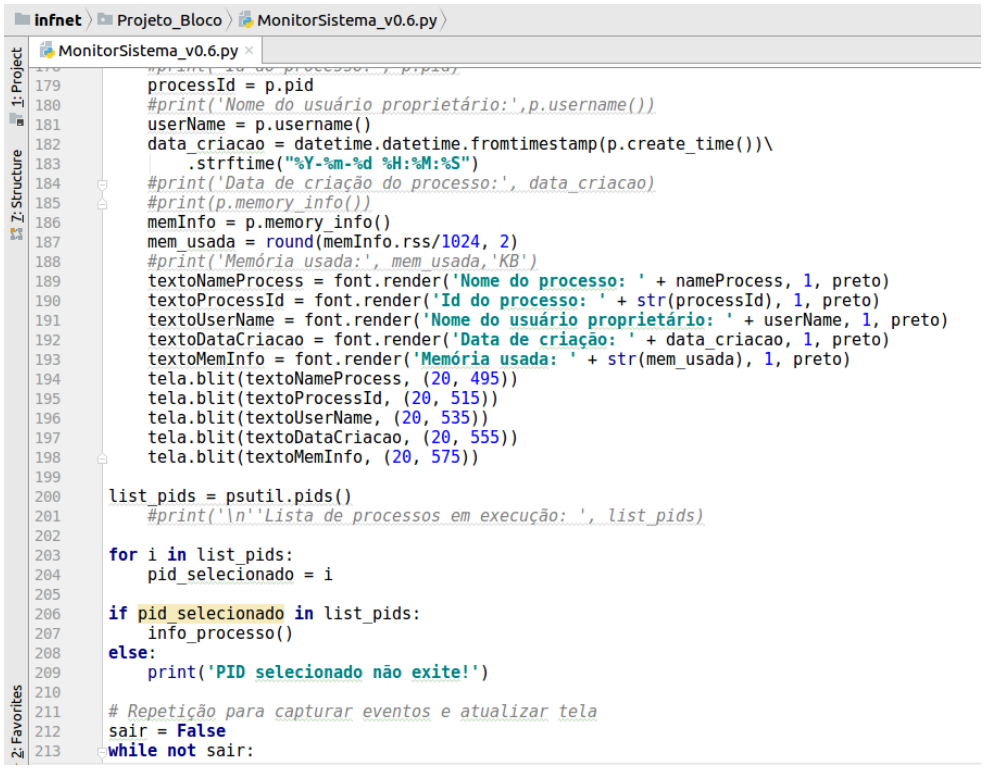



```
97      total = round(mem.total/(1024*1024*1024),2)
98      totalEmUso = round(mem.used/(1024*1024*1024),2)
99      textoMemTopo = font.render('Capacidade total MP: '
100                                + str(total) + ' GB', 1, branco)
101     textoBarra = font.render(str(totalEmUso) + ' GB', 1, branco)
102     tela.blit(textoMemTopo,(20,2*alturaTela/6 + 10))
103     tela.blit(textoBarra,(250,300))
104
105 # Mostra a capacidade do disco rígido em uso e total:
106 def mostraUsoDisco():
107     lista_info_particao = psutil.disk_partitions()
108     disco = psutil.disk_usage('.')
109     largBarraDisco = larguraTela-2*20
110     s3.fill(cinza)
111     pygame.draw.rect(s3, azul, (20,30,largBarraDisco,50))
112     tela.blit(s3, (0,3*alturaTela/6))
113     largBarraDisco = largBarraDisco*disco.percent/100
114     pygame.draw.rect(s3, vermelho, (20,30,largBarraDisco,50))
115     tela.blit(s3,(0,3*alturaTela/6))
116     total = round(disco.total/(1024*1024*1024),2)
117     discoUso = round(disco.used/(1024*1024*1024),2)
118     textoDiscoTopo = font.render('Capacidade total Disco Rígido: '
119                                  + str(total) + ' GB', 1, branco)
120     textoBarra = font.render(str(discoUso) + ' GB', 1, branco)
121     textoInfoParticao = font.render('| Dispositivo: ' + str(lista_info_particao[0].device)
122                                     + '  |  Sistema de Arquivo: '
123                                     + str(lista_info_particao[0].fstype), 1, branco)
124     tela.blit(textoInfoParticao, (350, 3*alturaTela/6 + 10))
125     tela.blit(textoDiscoTopo,(20, 3*alturaTela/6 + 10))
126     tela.blit(textoBarra,(25, 3*alturaTela/6 + 50))
127
128 # Mostra o ip da interface de rede local:
129 def mostraIp():
130     dicIpAdress = psutil.net_if_addrs()
131     s4.fill(branco)
132     tela.blit(s4,(0, 4*alturaTela/6))
133     textoIpAdress = font.render('Rede local Ip: '
134                                 + dicIpAdress['enp0s7'][0].address, 1, preto)
135     tela.blit(textoIpAdress,(20, 410))
136
137 def mostraInfoDir():
138     #s4.fill(branco)
139     #tela.blit(s4, (0, 4*alturaTela/5))
140
141     lista = os.listdir()
142     dic = {}
```

Resposta para os requisitos 3 e 4:

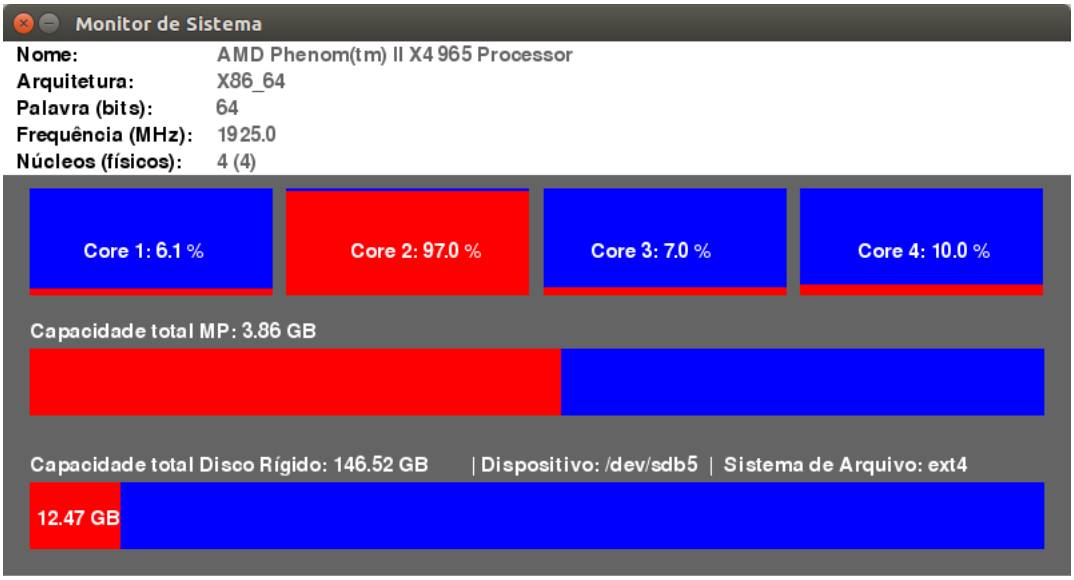


```
144         for i in lista:
145             if os.path.isfile(i):
146                 dic[i] = []
147                 dic[i].append(os.stat(i).st_size)
148                 dic[i].append(os.stat(i).st_ctime)
149                 dic[i].append(os.stat(i).st_mtime)
150 
151         tituloInfo = 'Arquivos e Diretórios:'
152         titulo_tamanho = '{:>5}'.format('Tamanho')
153         titulo_data_criacao = '{:>30}'.format('Data de Criação')
154         titulo_data_modificacao = '{:>38}'.format('Data de Modificação')
155         titulo = titulo_tamanho + titulo_data_criacao + titulo_data_modificacao
156         #print(titulo)
157         textoTituloInfo = font.render(tituloInfo, 1, preto)
158         tela.blit(textoTituloInfo, (20, 430))
159         textoTitulo = font.render(titulo, 1, preto)
160         tela.blit(textoTitulo, (20, 450))
161 
162         for i in dic:
163             kb = dic[i][0]/1024
164         tamanho = '{:>10}'.format(str('{:.2f}'.format(kb) + 'KB'))
165         time_create = '{:>30}'.format(time.ctime(dic[i][0]))
166         time_mod = '{:>30}'.format(time.ctime(dic[i][1]))
167         nomeArquivo = '{:>30}'.format(i)
168         textoArqDir = font.render(tamanho + time_create + time_mod
169                                   + nomeArquivo, 1, preto)
170         tela.blit(textoArqDir, (20, 470))
171 
172 def info_processo():
173         s5.fill(branco)
174         tela.blit(s5, (0, 5*alturaTela/6))
175         p = psutil.Process(pid_selecionado)
176         #print('\n''Nome do processo:', p.name())
177         nameProcess = p.name()
178         #print('Id do processo:', p.pid)
```



```
179         processId = p.pid
180         #print('Nome do usuário proprietário:',p.username())
181         userName = p.username()
182         data_criacao = datetime.datetime.fromtimestamp(p.create_time())\
183             .strftime("%Y-%m-%d %H:%M:%S")
184         #print('Data de criação do processo:', data_criacao)
185         #print(p.memory_info())
186         memInfo = p.memory_info()
187         mem_usada = round(memInfo.rss/1024, 2)
188         #print('Memória usada:', mem_usada,'KB')
189         textoNameProcess = font.render('Nome do processo: ' + nameProcess, 1, preto)
190         textoProcessId = font.render('Id do processo: ' + str(processId), 1, preto)
191         textoUserName = font.render('Nome do usuário proprietário: ' + userName, 1, preto)
192         textoDataCriacao = font.render('Data de criação: ' + data_criacao, 1, preto)
193         textoMemInfo = font.render('Memória usada: ' + str(mem_usada), 1, preto)
194         tela.blit(textoNameProcess, (20, 495))
195         tela.blit(textoProcessId, (20, 515))
196         tela.blit(textoUserName, (20, 535))
197         tela.blit(textoDataCriacao, (20, 555))
198         tela.blit(textoMemInfo, (20, 575))
199 
200 list_pids = psutil.pids()
201         #print('\n''Lista de processos em execução: ', list_pids)
202 
203 for i in list_pids:
204         pid_selecionado = i
205 
206 if pid_selecionado in list_pids:
207         info_processo()
208 else:
209         print('PID selecionado não exite!')
210 
211 # Repetição para capturar eventos e atualizar tela
212 sair = False
213 while not sair:
```

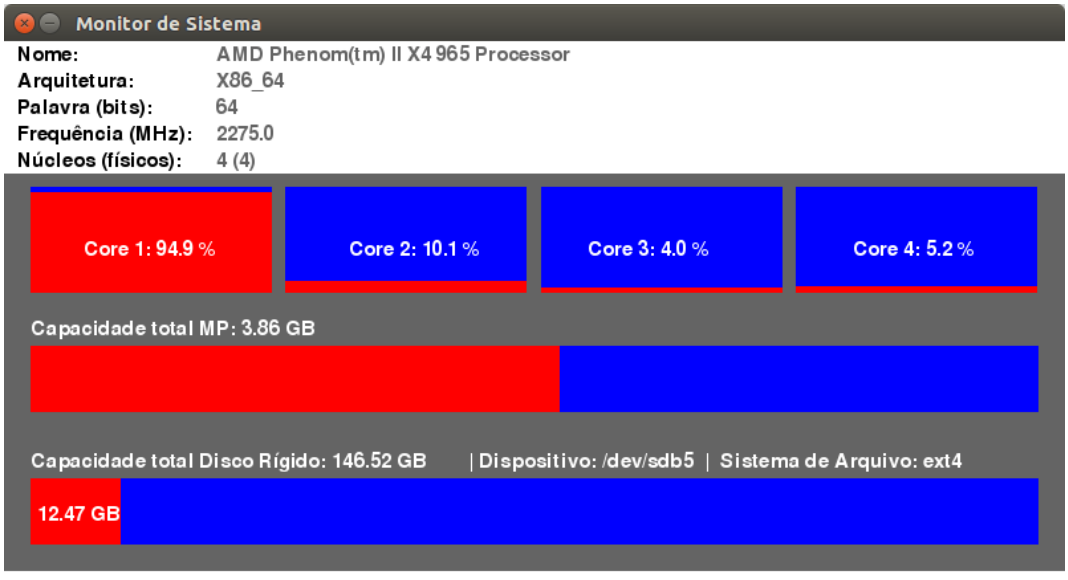
```
210
211 # Repetição para capturar eventos e atualizar tela
212 sair = False
213 while not sair:
214     # Checar os eventos do mouse aqui:
215     for event in pygame.event.get():
216         if event.type == pygame.QUIT:
217             sair = True
218
219     # Fazer a atualização a cada segundo:
220     if cont == 60:
221         mostraInfoCpu()
222         mostraUsoCpu(s0, listCpuPercent)
223         mostraUsoMemoria()
224         mostraUsoDisco()
225         mostraIp()
226         mostraInfoDir()
227         info_processo()
228         cont = 0
229
230     # Atualiza o desenho na tela
231     pygame.display.update()
232
233     # 60 frames por segundo
234     clock.tick(60)
235     cont = cont + 1
236
237 # Finaliza a janela
238 pygame.display.quit()
239 pygame.quit()
```



Rede local Ip: 192.168.88.254
Arquivos e Diretórios:

Tamanho	Data de Criação	Data de Modificação	
8.82KB	Wed Dec 31 23:30:33 1969	Fri Feb 16 16:24:17 2018	MonitorSistema_v0.6.py

Nome do processo: python
Id do processo: 7312
Nome do usuário proprietário: alexander
Data de criação: 2018-02-16 16:27:20
Memória usada: 41592.0



Rede local Ip: 192.168.88.254
Arquivos e Diretórios:

Tamanho	Data de Criação	Data de Modificação	
8.82KB	Wed Dec 31 23:30:33 1969	Fri Feb 16 16:24:17 2018	MonitorSistema_v0.6.py

Nome do processo: python
Id do processo: 7261
Nome do usuário proprietário: alexander
Data de criação: 2018-02-16 16:24:16
Memória usada: 41476.0

Bibliografia

PYTHON.ORG. datetime — Basic date and time types. Disponível em: https://docs.python.org/3/library/datetime.html. Acesso em: 10 fev 2018.

PYTHON.ORG. os.path — Common pathname manipulations. Disponível em: https://docs.python.org/3/library/os.path.html. Acesso em: 10 fev 2018.

PYTHON.ORG. psutil documentation. Disponível em: http://psutil.readthedocs.io/en/latest/. Acesso em: 10 fev 2018.

PYTHON.ORG. subprocess — Subprocess management. Disponível em: https://docs.python.org/3/library/subprocess.html?highlight=subprocess. Acesso em: 10 fev 2018.

PYTHON.ORG. time — Time access and conversions. Disponível em: https://docs.python.org/3/library/time.html. Acesso em: 10 fev 2018.