ESTI – ESCOLA SUPERIOR DA TECNOLOGIA DA INFORMAÇÃO
GRADUAÇÃO EM ENGENHARIA DE SOFTWARE
PROJETO DE BLOCO

Matéria
TESTE DE PERFORMANCE

Nome: Wellerson Carlos Amaral Moreira
Prof: Adriano Saad

Rio de Janeiro, 25 de maio de 2020.

ENTREGÁVEL

Objetivo desse entregável é mostrar de forma detalhada os principais componentes do computador e monitorar esses componentes. Os principais componentes que serão monitorado será CPU, memória ram e disco rígido. Algumas informações do computador será mostrada na tela como IP, nome do computador, sistema operacional, plataforma e bits do processador.

MÓDULOS E FERRAMENTAS

Foram usados 3 módulos e irei explicar algumas das ferramentas ao decorrer que for falando dos módulos, módulo pygame foi utilizado para os retângulos desenhado na tela. Pygame permite fazermos alguns desenhos geométricos na tela conforme eu disse, além disso também permite algumas escritas. Nesse entregável irá ter 6 retângulos com alguns dados que irei falar mais a frente. Psutil um módulo do python que ajuda a pegar algumas informações de alguns componentes do computador, tais como CPU, disco rígido e memória ram. Esse último módulo usado chamado platform foi usado para obter algumas informações do computador, tais como nome, plataforma, bits do processador e sistema operacional.

PYGAME

Pygame foi utilizado para desenhar os retângulos na tela, ele permite desenhos geométricos com coordenada x, y, tamanho e altura. além disso ele permite você colocar texto na tela que está desenhando. Nesse entregável basicamente pygame foi utilizado para visualização, nada além disso vamos dizer que ele é nosso frontend.

PSUTIL

PSUtil utilizamos métodos prontos que ele nos fornece para obter algumas informações de alguns componentes de nosso computador, com ele pude fazer chamadas de métodos que nos fornece informação da CPU, memória ram e disco rígido.

Logo no primeiro texto com o primeiro retângulo no lado superior esquerdo, temos informações da CPU. Você irá poder ver os GHz do seu processador e quantos porcentos ele está utilizando de sua máquina.

Segundo processo seria no lado superior direito, você também irá ver um retângulo mostrando o total de GB que você tem e o quantos de GB que sua memória ram está usando.

Logo abaixo desses dois item vai ter informações do seu disco rígido, total de GB que tem em sua máquina e quantos está usando.

Todos esses itens acima de informação é graças ao PSUtil, com ele posso buscar informações com métodos prontos para verificar essas informações do seu computador para saber mais do PSUtil irei deixar aqui a documentação, recomendo a leitura!

PLATFORM

Módulo para obter algumas informações do computador, este módulo já vem com o python3 então fazer a importação dele fica de fácil acesso. Com ele você pode obter informações como nome do computador, sistema operacional, bits de processador, plataforma e etc. Consigo obter essas informações a partir de métodos prontos, recomendo a leitura da documentação.

CPUINFO

CPU-INFO é um módulo que nos ajuda a ganhar mais informações da CPU, conseguindo trazer informações valiosas da cpu. Com ele podemos ver quantos núcleos tanto físico quanto lógico nosso processador tem, muito além disso podemos trazer também informações da arquitetura, palavra (bits) e nome do processador. Podemos ter mais informações irei deixar a documentação se quiser obter informações além.

TELAS

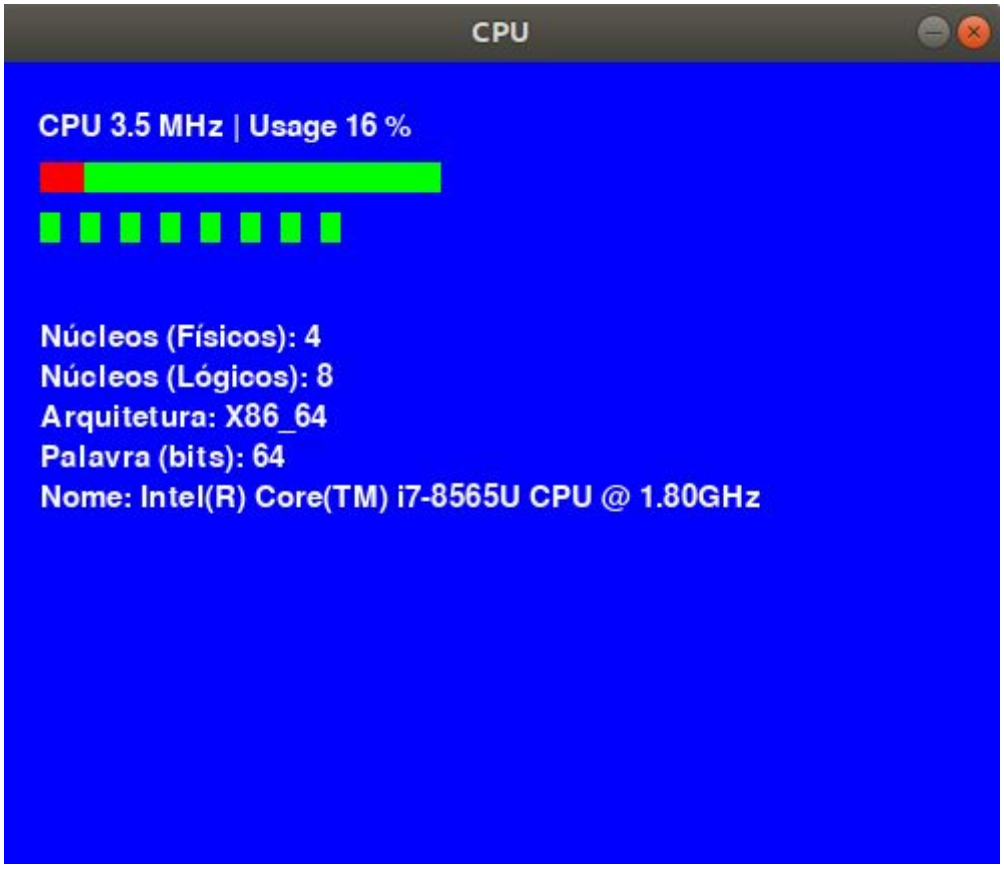
CONTROLE DAS TELAS

Antes de começar a falar das telas em si, quero mencionar de como controlá-las. Assim que entrar no sistema a primeira tela a ser aberta irá ser a tela de cpu, com isso o controle de tela é bem simples. Ao clicar com a seta do teclado para a direito ele muda de tela, (com isso ele irá para a tela de memória) ao clicar para a

esquerda ele mudar para a tela anterior. Quero levantar que a primeira tela irá ser da cpu e a última irá ser a de IP. Ao clicar no espaço do teclado ele irá para informações gerais do sistema.
obs: Ao entrar na informações gerais do sistema você pode voltar de tela clicando para a esquerda na seta do teclado.

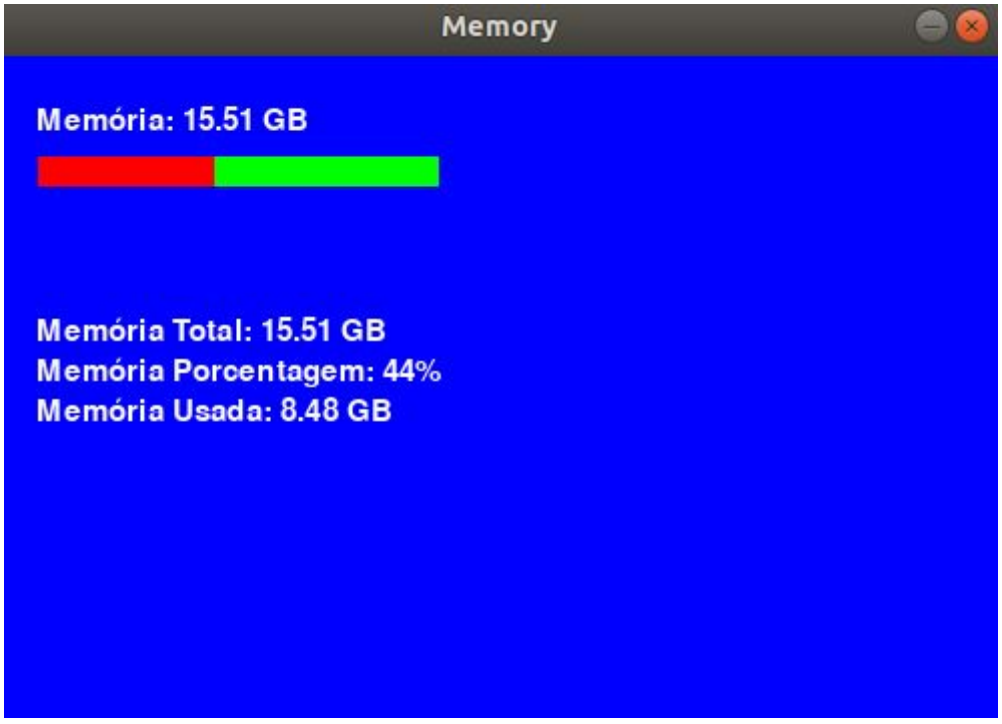
TELA CPU

A tela de cpu é composta de algumas informações importante da cpu do seu computador, ele mostra de maneira detalhada a porcentagem que você está usando da sua cpu e quantos GHz o processador está trabalhando. Além disso também mostra uma barra de informação com as cores verde e vermelho, a cor verde mostra o quanto ainda você tem livre para usar, já a cor vermelha mostra o quanto o processador está usando naquele momento. Logo abaixo você tem alguns blocos verdes indicando o quantos de núcleo lógicos seu processador tem. Contudo além dessa informações temos algumas outras de núcleo físico e lógico, arquitetura, palavras (bits) e o nome do processador. Confira na imagem abaixo.



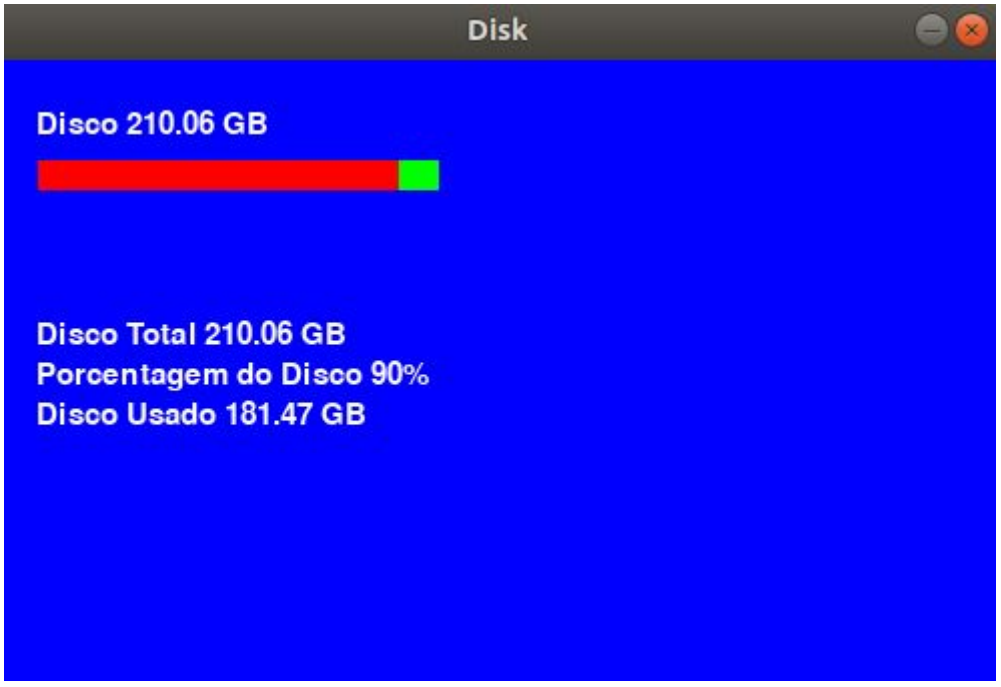
TELA MEMÓRIA

Todo computador tem uma memória não é mesmo ? Essa tela é dedicada para o monitoramento da memória como diz o título. temos informações bem úteis aqui, essa tela é bem parecido com a tela da cpu ela também vem com uma barra com a cores verde e vermelho, verde indica o quanto porcento de memória você ainda tem livre, já o vermelho indica o quanto você está gastando no momento. Nessa tela você tem informações em GB, o total de memória que você possui, quantos porcentos está utilizando no momento e o quanto de memória você está usando no momento. Confira na imagem.



TELA DISCO

Tela de disco rígido tem temos informações em GB do seu disco rígido, ele mostra uma barra de indicação como todas as outras e ter dar algumas informações de quanto GB você tem na sua máquina, quantos porcentos esse disco está e quantos GB de espaço usado em seu computador, irei deixar uma imagem para mais detalhes.

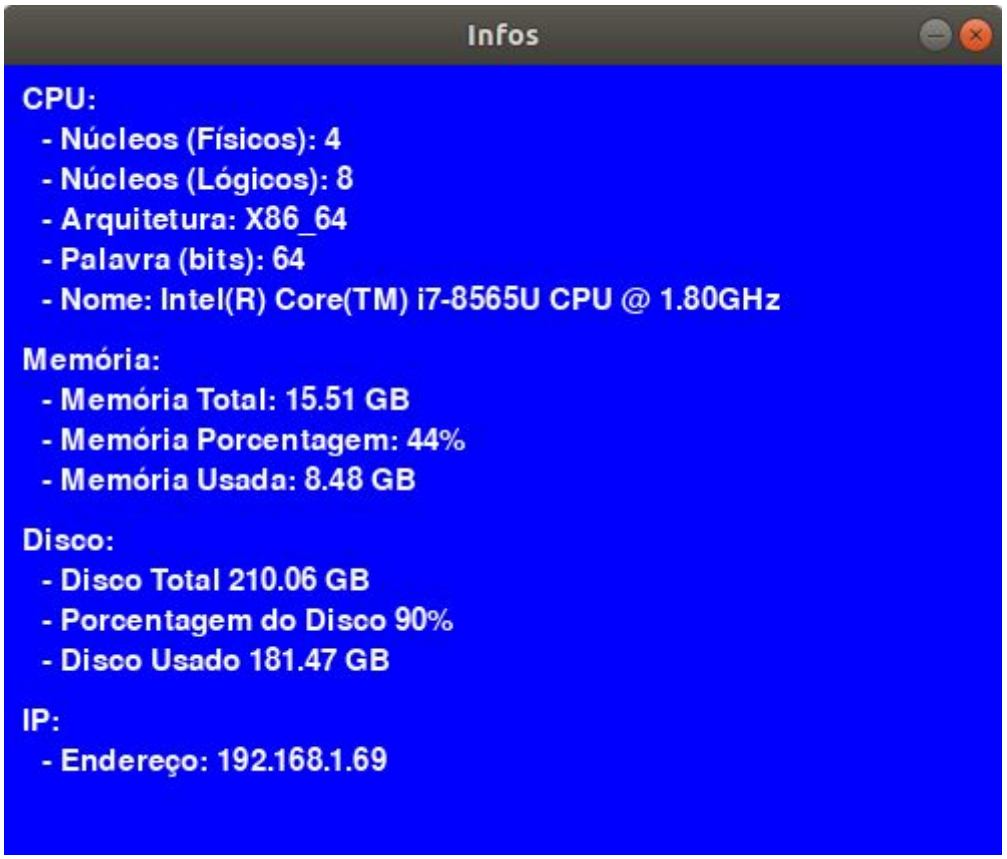


TELA IP

IP fica algumas informações de rede da máquina, essa tela é composta de 4 informações. Com ela você irá conseguir ver seu endereço local, endereço IP, endereço mask e endereço broadcast. Imagem abaixar para mais detalhes do layout.

TELA INFOS

Essa tela é composta de informações variadas, ela vai te dar um resumo geral de todas as tela, aqui você pode ver informações da cpu, memória, disco, e ip. Irei deixar uma imagem para mais detalhe.

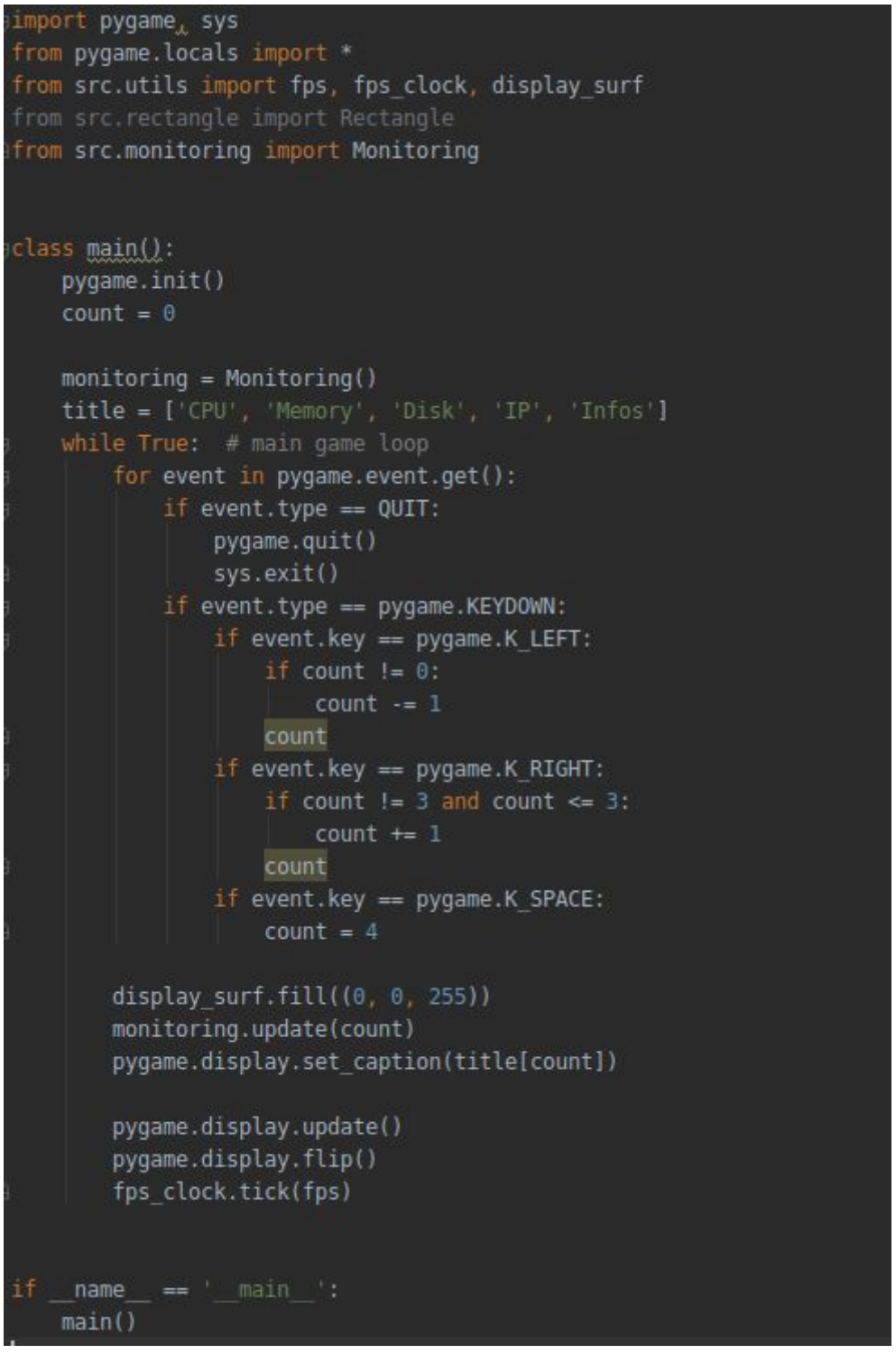


PARTE PARA PROGRAMADORES

MAIN

Esse programa foi feita orientada a objetos com python, irei explicar como criei algumas coisas.

Logo de início queria falar como ele está sendo iniciado, temos uma class main iniciando o pygame e chamando uma class chamada monitoring.



```python
import pygame, sys
from pygame.locals import *
from src.utils import fps, fps_clock, display_surf
from src.rectangle import Rectangle
from src.monitoring import Monitoring


class main():
    pygame.init()
    count = 0

    monitoring = Monitoring()
    title = ['CPU', 'Memory', 'Disk', 'IP', 'Infos']
    while True:  # main game loop
        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT:
                    if count != 0:
                        count -= 1
                    count
                if event.key == pygame.K_RIGHT:
                    if count != 3 and count <= 3:
                        count += 1
                    count
                if event.key == pygame.K_SPACE:
                    count = 4

        display_surf.fill((0, 0, 255))
        monitoring.update(count)
        pygame.display.set_caption(title[count])

        pygame.display.update()
        pygame.display.flip()
        fps_clock.tick(fps)


if __name__ == '__main__':
    main()
```

Temos um while sempre TRUE e com um for dentro dele para pegar alguns eventos do pygame disponibiliza para a gente, em primeiro momento é confuso mas você irá se acostumar com essa estrutura. logo abaixo do for faço alguns updates,

como estamos sempre tendo que dar um update para poder pintar as informações atualizada. Esse display_surf.fill ele pinta a tela de azul sempre e faz um update logo em seguida, no update do monitoring passo sempre um count que consegue controlar minhas surfaces e o título, esse count atualiza conforme clica na seta ou no espaço do teclado. (Cá entre nós, acho que isso é uma gambiarra)

MONITORING

Está class fica toda a logica do programa, é aqui que ele monta os retângulos e faz o texto aparece na tela.

```
class Monitoring():
    def __init__(self):
        ########################################################################
        # CPU
        self.cpu = CPU()
        self.surfCpu = Surface(0, 0, 500, 400)
        self.textCpu = Text(19, 25, self.surfCpu.getSurface(), 15)
        self.rectCpu = Rectangle(20, 50, 200, 15, self.surfCpu.getSurface(), GREEN)
        self.rectCpuUsage = Rectangle(20, 50, 200, 15, self.surfCpu.getSurface(), RED)
        # logic core cpu
        self.xCores = 0
        self.xCoresInside = 0
        self.rectCores = []
        for core in range(self.cpu.getCoresLogical()):
            self.xCores += 20
            if self.xCores > 180:
                self.xCoresInside += 20
                self.rectCores.append(Rectangle(self.xCoresInside, 95, 10, 15, self.surfCpu.getSurface(), GREEN))
            self.rectCores.append(Rectangle(self.xCores, 75, 10, 15, self.surfCpu.getSurface(), GREEN))
        # infos cpu
        self.textCore = Text(20, 130, self.surfCpu.getSurface(), 15)
        self.textCoreLogical = Text(20, 150, self.surfCpu.getSurface(), 15)
        self.textArch = Text(20, 170, self.surfCpu.getSurface(), 15)
        self.textBits = Text(20, 190, self.surfCpu.getSurface(), 15)
        self.textBrand = Text(20, 210, self.surfCpu.getSurface(), 15)
        ########################################################################
        ########################################################################
        # MEMORY
        self.memory = Memory()
        self.surfMemory = Surface(0, 0, 500, 400)
        self.rectMemory = Rectangle(20, 50, 200, 15, self.surfMemory.getSurface(), GREEN)
        self.rectMemoryUsage = Rectangle(20, 50, 200, 15, self.surfMemory.getSurface(), RED)
        # memory info
        self.textMemory = Text(19, 25, self.surfMemory.getSurface(), 15)
        self.textMemoryTotal = Text(19, 130, self.surfMemory.getSurface(), 15)
        self.textMemoryPercent = Text(19, 150, self.surfMemory.getSurface(), 15)
        self.textMemoryUsage = Text(19, 170, self.surfMemory.getSurface(), 15)
        ########################################################################
        ########################################################################
        # DISK
        self.disk = Disk()
        self.surfDisk = Surface(0, 0, 500, 400)
        self.rectDisk = Rectangle(20, 50, 200, 15, self.surfDisk.getSurface(), GREEN)
        self.rectDiskUsage = Rectangle(20, 50, 200, 15, self.surfDisk.getSurface(), RED)
        # disk info
```

Nela temos um método chamado update, ele serve para dar um update na tela e controla o array de surfaces com o count com isso ele consegue trazer as informações atualizadas, já que é um monitoramento e sem vai está mudando alguns dados.

```
def update(self, count):
    self.arrSurface[count].draw()
    ################################################################################
    # cpu
    percentCPU = round((self.cpu.getPercentUsage() * 200) / 100)
    self.rectCpu.draw()
    self.rectCpuUsage.draw()
    self.rectCpuUsage.update(percentCPU)
    self.textCpu.draw(f'CPU {self.cpu.getCpu()} MHz | Usage {percentCPU} %')
    # logic core cpu
    for core in range(self.cpu.getCoresLogical()):
        self.rectCores[core].draw()
    # infos cpu
    self.textCoreLogical.draw(f'Núcleos (Lógicos): {self.cpu.getCoresLogical()}')
    self.textCore.draw(f'Núcleos (Físicos): {self.cpu.getCores()}')
    self.textArch.draw(f'Arquitetura: {self.cpu.getArch()}')
    self.textBits.draw(f'Palavra (bits): {self.cpu.getBits()}')
    self.textBrand.draw(f'Nome: {self.cpu.getBrand()}')
    ################################################################################
    ################################################################################
```

Você pode ver que chamamos duas classes chamada Rectangle e Text, a classe Rectangle irei falar dela mais a frente mas basicamente ele serve para fazer um retângulo. Já a classe Text serve para escrever algo na tela.

RECTANGLE

Classe rectangle serve para desenhar um retângulo na tela, você para algumas informações como x, y, tamanho, largura e cor, Logo após isso você irá ter um retângulo porém ele não vai escrever na tela, você deve usar o método draw para ele desenhar seu retângulo na tela.

```python
class Rectangle(pygame.sprite.Sprite):
    def __init__(self, x, y, w, h, color):
        self.x = x
        self.y = y
        self.w = w + 200
        self.h = h
        self.color = color

    # Draws the rect
    def draw(self, width=False):
        if width:
            self.w = (width * 300) / 100
        self.rect = pygame.Rect(self.x, self.y, self.w, self.h)

        # Draws rect
        pygame.draw.rect(display_surf, self.color, self.rect)
```

Para você desenhar um retângulo basta você importar ele e passar os valores

```python
self.CPU = Rectangle(50, 50, 100, 30, GREEN)
```

logo após isso você pode desenhar ele na tela

```python
def update(self):
    # CPU
    self.CPU.draw()
```

TEXT

Essa classe é simples, ela tem o objetivo de pintar um texto na tela. Ela recebe x, y e tamanho da fonte. Lembrando a fonte por padrão é 20.

```
class Text():
    def __init__(self, x=25, y=25, font_size=20):
        self.x = x
        self.y = y

        self.font = pygame.font.Font('freesansbold.ttf', font_size)

    # Displays the current score on the screen
    def display(self, text):
        self.text = text
        result_surf = self.font.render(self.text, True, WHITE)
        rect = result_surf.get_rect()
        rect.topleft = (self.x, self.y)
        display_surf.blit(result_surf, rect)
```

O método display fica de receber o texto para colocar dentro do render, para escrever algo é bem simples.

Primeiro é preciso fazer o importe passando os argumentos x e y, a fonte por padrão é 20 como eu tinha dito

```
self.CPUText = Text(50, 50 - 25)
```

Depois disso é só usar o método display para passar o texto que você queira jogar na tela.

```
def update(self):
    # CPU
    self.CPUText.display(f'CPU {freq()} GHZ')
```

sempre dentro do update porque as informações pode atualizar.

SURFACE

Classe surface foi criada para manter uma lógica no código, eu posso controlar qual surface eu quero desenhar na tela, por isso eu criei essa classe, vamos ver como ela funciona.

Ao instanciar essa classe você poderá para o construtor eixo x e y com a largura e altura, ela tem 2 métodos um getSurface para retornar sua surface e um para

desenhar. o método draw sempre vai desenhar dentro do display do pygame, nesse caso o display main

```python
import pygame
from src.utils import display_surf, BLUE


class Surface():
    def __init__(self, x, y, width, height):
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.surface = pygame.Surface((self.width, self.height))

    def getSurface(self):
        return self.surface

    def draw(self, color=BLUE):
        display_surf.blit(self.surface, (self.x, self.y))
        self.surface.fill(color)
```

LÓGICA DE SLIDE

Essa lógica é bem simples no caso eu controlo qual surface eu quero desenhar na tela, aqui tenho um array de surface

```python
# arrays surface
self.arrSurface = [self.surfCpu, self.surfMemory, self.surfDisk, self.surfIp, self.surfInfo]
```

no método update na classe monitoring recebemos um argumento chamado de count, ele controla qual surface eu desenho na tela

```python
def update(self, count):
    self.arrSurface[count].draw()
```

A classe main passa um count atualizado para ele.

FINAL

Por final, eu irei mostrar o layout que eu desenhei com essas classes