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, 23 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.

IP

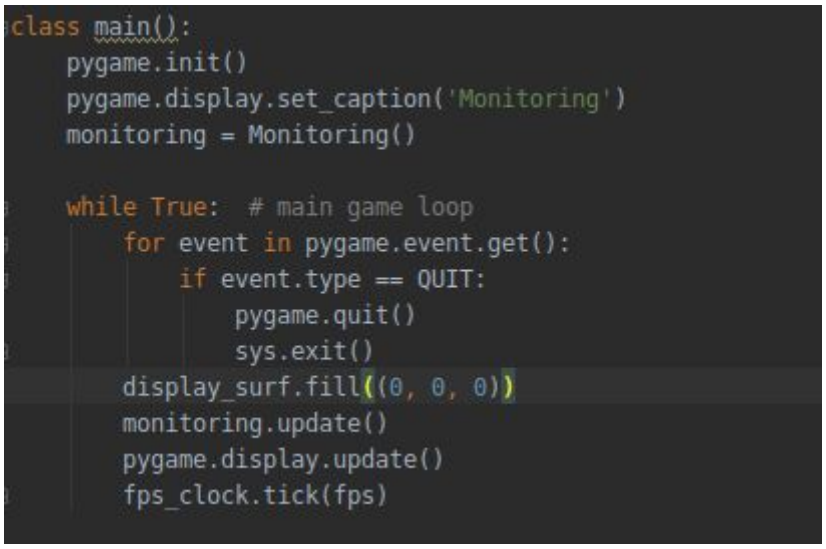
IP no layout fica junto com as informações do computador, IP tem uma grande responsabilidade para a comunicação com outros dispositivos através de protocolos. Ele é importante pois se conecta a uma rede de computador e utiliza o protocolo de internet para comunicação.

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.



```
class main():
    pygame.init()
    pygame.display.set_caption('Monitoring')
    monitoring = Monitoring()

    while True:  # main game loop
        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
                sys.exit()
        display_surf.fill((0, 0, 0))
        monitoring.update()
        pygame.display.update()
        fps_clock.tick(fps)
```

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 preto sempre e faz um update logo em seguida. (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.

```python
class Monitoring(pygame.sprite.Sprite):
    def __init__(self):
        # CPU
        self.CPUtotalUsage = round(percentage().user + percentage().system)

        self.CPUText = Text(50, 50 - 25)
        self.CPU = Rectangle(50, 50, 100, 30, GREEN)
        # CPU Usage
        self.CPUUsageText = Text(50, 90)
        self.CPUUsage = Rectangle(50, 50, int(self.CPUtotalUsage), 30, RED)

        # Memory
        self.memory = Rectangle(450, 50, 100, 30, GREEN)
        self.memoryText = Text(450, 50 - 25)
        # Memory usage
        self.memoryUsage = Rectangle(450, 50, memory_usage_percent(), 30, RED)
        self.memoryTextUsage = Text(450, 90)

        # Disk
        self.disk = Rectangle(50, 200, 100, 30, GREEN)
        self.diskText = Text(50, 170)
        # disk usage
        self.diskUsage = Rectangle(50, 200, disk_percent(), 30, RED)
        self.diskTextUsage = Text(50, 240)
        # info ip
        self.ip = Text(50, 400)
        # systen
        self.platform_name = Text(50, 430)
        self.platform_platform = Text(50, 460)
        self.platform_system = Text(50, 490)
        self.platform_processor = Text(50, 520)
```

Nela temos um método chamado update, ele serve para dar um update na tela e vim com a informações atualizada, já que é um monitoramento e sem vai está

mudando alguns dados.

```
def update(self):
    # CPU
    self.CPU.draw()
    self.CPUText.display(f'CPU {freq()} GHZ')
    self.CPUUsage.draw(round(percentage().user + percentage().system))
    self.CPUUsageText.display(f'CPU Usage {round(percentage().user + percentage().system)} %')
    # Memory
    self.memory.draw()
    self.memoryText.display(f'Memory Total {Mem()} GB')
    self.memoryUsage.draw(memory_usage_percent())
    self.memoryTextUsage.display(f'Memory Usage {memory_usage()} GB')
    # Disk
    self.disk.draw()
    self.diskText.display(f'Disk Total {disk_total()} GB')
    # Disk Usage
    self.diskUsage.draw(disk_percent())
    self.diskTextUsage.display(f'Disk Usage {disk_usage()} GB')
    # IP
    self.ip.display(f'IP config {Ip()}')
    # system
    self.platform_name.display(f'Nome da plataforma: {platform_name()}')
    self.platform_system.display(f'Sistema Operacional: {platform_system()}')
    self.platform_platform.display(f'Plataforma: {platform_platform()}')
    self.platform_processor.display(f'Bits do processador: {platform_processor()}')
```

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.

```python
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

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

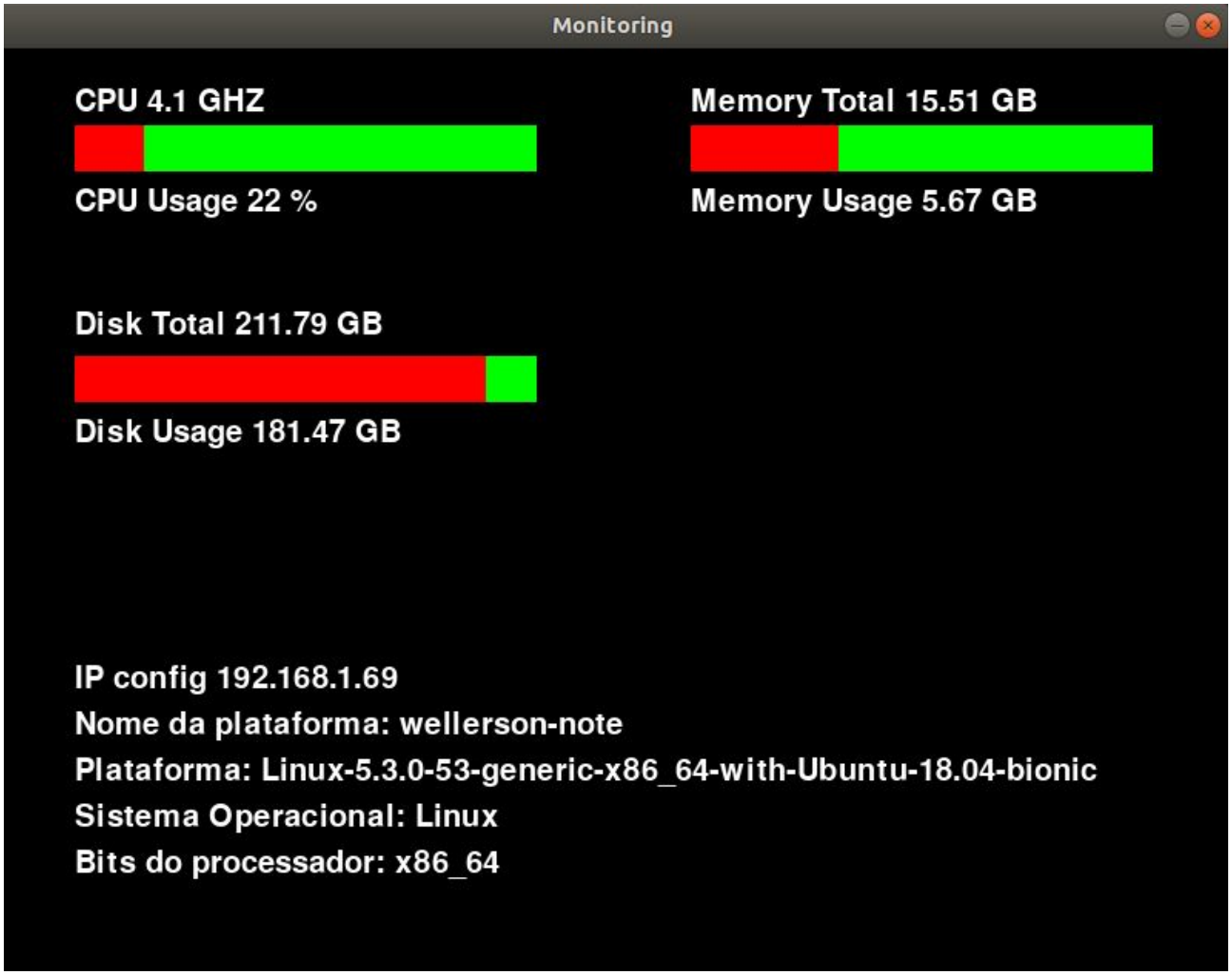
Depois disso é só usar o método display para passar o texto que você queira jogar na tela.

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

sempre dentro do update porque as informações pode atualizar.

FINAL

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


Monitoring
CPU 4.1 GHZ
CPU Usage 22 %
Memory Total 15.51 GB
Memory Usage 5.67 GB
Disk Total 211.79 GB
Disk Usage 181.47 GB
IP config 192.168.1.69
Nome da plataforma: wellerson-note
Plataforma: Linux-5.3.0-53-generic-x86_64-with-Ubuntu-18.04-bionic
Sistema Operacional: Linux
Bits do processador: x86_64