UNIVERSIDADE CATÓLICA DE PELOTAS

ENGENHARIA DE COMPUTAÇÃO

PROJETO INTEGRADOR VI

ÍCARO GONÇALVES SIQUEIRA

MONITORAMENTO E ARMAZENAMENTO DE DADOS - 1ª ETAPA

23/09/2020

I. INTRODUÇÃO

A meta desta disciplina será seguir a linha de trabalho de Projeto Integrador 5 (PI5), ministrado pelo professor Adenauer no semestre passado, e adicionar a implementação de Banco de Dados, que está sendo ministrada no atual semestre pelo professor Marcos. Com esse objetivo em vista, o projeto consistirá na programação de um script que colete algumas informações relativas ao monitoramento do software e/ou do hardware e armazená-los em um banco de dados relacional.

Em PI5 foi vista a importância do monitoramento de um sistema em tempo real, no qual poderia ser usado para alertar uma emergência de funcionamento incorreto através de uma rede, além de dar a possibilidade de observar o comportamento recente de um dispositivo. Este tipo de projeto acaba sendo útil para servidores ou outros hardwares sem acesso manual frequente, mas apesar de sua utilidade foi notado um problema na maneira que o utilizamos.

Pelo fato de não termos usado um banco de dados, as coletas eram postadas na nuvem diretamente e algumas informações acabavam sendo perdidas. Os Valores antigos eram substituídos por novos, pois a visualização era feita por gráficos que, para poderem ser visualizados com eficiência, acabavam limitando a quantidade de dados que poderiam ser plotados.

Com a implementação do Banco de Dados em PI6 podemos armazenar uma grande quantidade de dados sem a necessidade de descartar dados antigos, além de ter um fácil acesso e visualização das informações desejadas, pois pode-se escolher qualquer parâmetro na consulta do dados do banco.

II. METODOLOGIA

Na etapa da coleta dos dados foi usado como base o script feito na disciplina PI5, que foi programado na linguagem Python 3. Tal linguagem foi recomendada por ambos professores de PI5 e PI6, por ser mais simples e direta, e ser a mais utilizada para a programação de scripts deste tipo. As mudanças no programa foram basicamente para implementação do banco de dados e outras mudanças estéticas.

A. Bibliotecas

- psutil: a biblioteca psutil foi a primeira e mais completa solução para coleta de informações em python encontrada. Além de dar a opção de uma grande quantidade de dados para coletar, sua documentação é bastante direta e simples, facilitando seu uso.

- warnings: esta biblioteca se fez necessária para melhor visualização das saídas do script. A biblioteca psutil teve uma incompatibilidade com a versão do sistema operacional usado, gerando diversos avisos durante sua execução. Os avisos não eram prejudiciais à coleta, então foi decidido apenas ignorá-los com a biblioteca warnings.

- pyspectator: pyspectator foi usada apenas para finalidade estética, sendo responsável por escrever o modelo do hardware na saída das informações.

- mysql.connector: Biblioteca escolhida para a comunicação do script com o banco de dados, a mysql.connector foi uma das bibliotecas propostas e cumpriu seu papel com maestria. Foi a única dos exemplos a não proporcionar nenhuma dificuldade em se comunicar com o BD.

- time: Biblioteca do python responsável pela coleta da data e hora do sistema. Além de transformar a data no formato timestamp, usada no banco de dados.

- subprocess: a biblioteca subprocess foi usada para rodar comandos bash diretamente de dentro do script, isso foi necessário pois não havia uma solução em python que fizesse a coleta de algumas informações. Abaixo os comandos usados:

    - nvidia-settings: Este comando foi o principal motivo pela necessidade do uso da biblioteca subprocess. Não foi encontrada nenhuma alternativa para coleta das informações da gpu sem que fosse pelo comando do driver proprietário da própria.

- smartctl: Comando que possibilitou o monitoramento das temperaturas dos discos de armazenamento, também sendo uma das únicas e mais simples maneiras encontradas para tal ação.

- lshw: Esta é uma ferramenta que informa inúmeras informações tanto de hardware quanto de software. Se mostrou um comando bastante útil e completo, mas neste script foi usada apenas de forma estética, da mesma forma que a biblioteca pyspectator.

B. Dados Monitorados

A escolha de quais dados seriam monitorados foi feita baseada em dois fatores. Primeiro foram escolhidos os dados que possuem uma maior variância no uso do computador, segundo foram escolhidos os dados que poderiam de alguma forma indicar um comportamento prejudicial para o sistema ou para os componentes.

Da CPU foram coletados as informações de Ocupação total do processador, Frequência de trabalho e Temperatura. Na GPU os dados coletados foram as mesmas da CPU além do uso de sua memória dedicada. Foi coletado também o uso da Memória RAM, dividido em memória usada e memória livre, para mais possibilidades de consulta. Dos discos foram coletadas as Temperaturas tanto do HDD quando do SSD. Por fim, foi também coletada a quantidade dados por segundo a rede está usando.

Informações de porcentagem de uso são úteis para saber se é possível ou não o início de novos processos, além de ser um complemento pra saber se as temperaturas correspondem ao nível de uso ou se é algum problema físico.

As coletas de frequência de trabalho e de uso da rede podem servir tanto para monitorar a intensidade do uso do dispositivo quando para ter uma ideia de que tipo de aplicação está sendo rodada.

Os dados de temperatura são importantes, principalmente no uso de um notebook, pois é um hardware com menor capacidade de refrigeramento por ser mais compacto. Com isso as temperaturas podem mais facilmente chegar a níveis prejudiciais aos componentes. Já os dados de RAM são provavelmente os mais importantes em relação à estabilidade do software e devem receber a devida atenção para que não cause congelamentos no sistema.

C. Código de Monitoramento

- **Conexão do Banco de Dados**

Na etapa de conexão com o banco de dados foi usado o comando “try” para tentar conectar e parar o código caso ocorra algum erro na conexão. Se a conexão for feita com sucesso um loop de coletas será iniciado.

```
sleep = 60 # Intervalo em segundos de cada postagem

# Tenta iniciar conexão com o banco de dados
try:
    connection = mysql.connector.connect(host='localhost', database='pivi', user='
        icaro', password='')

    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Conectado ao Servidor MySQL versão ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("Você está conectado ao banco de dados: ", record)

# Retorna mensagem de erro caso conexão falhe
except Error as e:
    print("Erro conectando ao MySQL", e)

finally:

    cont = 0
    while cont < 60: # Numero de ciclos de leituda das informações
        cont = cont + 1
```

Na etapa de leituras foram apenas implementados os códigos anteriormente explicados na seção II.A. É feita uma pausa de 1 segundo para a medição do uso da rede, pois o valor medido é o total de dados usados, e com esta pausa pode-se obter a diferença do valor em um segundo. No fim é obtido o horário e a data em forma de timestamp para documentação no banco de dados.

Nos tratamentos de valores foram mantidas as formatações estéticas usadas em PI5 e foram adicionadas conversões para float e int, formatos utilizados nos campos do banco de dados. Esta etapa é importante pois alguns métodos de coleta geram saídas em formato binário, impossibilitando que o valor possa ser manipulado ou imprimido.

- **Leitura das Informações**

```
########################## Leitura das informações ###################################

#batt = psutil.sensors_battery()

#chrg = subprocess.check_output(["sudo", "dmidecode", "-t", "22"])

#NET

old_value = psutil.net_io_counters().bytes_sent + psutil.net_io_counters().bytes_recv
time.sleep(1)
new_value = psutil.net_io_counters().bytes_sent + psutil.net_io_counters().bytes_recv
net_sts = new_value - old_value

#CPU

cpu_percent = psutil.cpu_percent(interval=1)
cpu_freq = psutil.cpu_freq(percpu=False)
cpu_temp = psutil.sensors_temperatures(fahrenheit=False)

#RAM

mem = psutil.virtual_memory()

ram = subprocess.check_output(["sudo", "lshw", "-short", "-C", "memory"])

#HDD

hd_tmp = subprocess.check_output(["sudo", "smartctl", "-A", "/dev/sda"])

hd = subprocess.check_output(["sudo", "smartctl", "-i", "/dev/sda"])

#SSD

ssd_tmp = subprocess.check_output(["sudo", "smartctl", "-A", "/dev/nvme0"])

ssd = subprocess.check_output(["sudo", "smartctl", "-i", "/dev/nvme0"])

#GPU

gpu = subprocess.check_output(["nvidia-settings", "-q", "gpus", "-t"])

gpu_temp = subprocess.check_output(["nvidia-settings", "-q", "gpucoretemp", "-t"])
gpu_freq = subprocess.check_output(["nvidia-settings", "-q", "GPUCurrentClockFreqs", "-t"])
gpu_mem = subprocess.check_output(["nvidia-settings", "-q", "UsedDedicatedGPUMemory", "-t"])
gpu_percent = subprocess.check_output(["nvidia-settings", "-q", "GPUUtilization", "-t"])

data = strftime("%Y-%m-%d", localtime())
hora = strftime("%H:%M:%S", localtime())

datahora = strftime("%Y-%m-%d %H:%M:%S", localtime())
datahora = time.mktime(datetime.datetime.strptime(datahora,"%Y-%m-%d %H:%M:%S").timetuple())
```

- **Tratamento dos Valores**

```
######################## Tratamento de valores ################################

#Bateria

#chrg = chrg.decode("utf-8")
#chrg = chrg.split("\n")
#chrg1 = chrg[12]
#chrg2 = chrg[13]
#dummy, chrg1 = chrg1.split(":")
#dummy, chrg2 = chrg2.split(":")
#chrg = chrg1.strip(" ") + chrg2

#NET

net_sts = net_sts/1000

#HDD

hd_tmp = hd_tmp.decode("utf-8")
hd_tmp = hd_tmp.split("\n")
hd_tmp = hd_tmp[19]
dummy, hd_tmp = hd_tmp.split("-")
hd_tmp = hd_tmp.strip(" ")

hd_tmp = int(hd_tmp)

hd = hd.decode("utf-8")
hd = hd.split("\n")
hd1 = hd[4]
hd2 = hd[8]
dummy, hd1 = hd1.split(":")
dummy, hd2 = hd2.split("[")
hd = hd1.strip(" ") + " " + hd2.strip(" ]")

#SSD

ssd_tmp = ssd_tmp.decode("utf-8")
ssd_tmp = ssd_tmp.split("\n")
ssd_tmp = ssd_tmp[6]
dummy, ssd_tmp = ssd_tmp.split(":")
ssd_tmp = ssd_tmp.strip(" Celscius")

ssd_tmp = int(ssd_tmp)

ssd = ssd.decode("utf-8")
ssd = ssd.split("\n")
ssd = ssd[4]
dummy, ssd = ssd.split(":")
ssd = ssd.strip(" ")
```

```
#RAM

mem_free = float(round(mem.available/1000000,2))
mem_used = float(round(mem.used/1000000,2))

ram = ram.decode("utf-8")
ram = ram.split("\n")
ram1 = ram[5]
ram2 = ram[4]
dummy, ram1 = ram1.split("memory")
dummy, ram2 = ram2.split("memory")
ram = ram1.strip(" ") + " + " + ram2.strip(" ")

#CPU

cpu_percent = float(cpu_percent)

cpu_freq = float(round(cpu_freq.current,2))

names = list(cpu_temp.keys())
if names[2] in cpu_temp:
    for entry in cpu_temp[names[2]]:
        #print("%s %s°C" % (entry.label, entry.current))
        break
cpu_temp = entry.current

cpu_temp = int(cpu_temp)

#GPU

gpu = gpu.decode("utf-8")
gpu = gpu.split("\n")
gpu = gpu[2].split("(")
gpu = gpu[1].strip(")")

gpu_temp = gpu_temp.decode("utf-8")
gpu_temp = gpu_temp.split("\n")

gpu_temp = int(gpu_temp[0])

gpu_freq = gpu_freq.decode("utf-8")
gpu_freq = gpu_freq.split("\n")
gpu_freq = gpu_freq[0].split(",")

gpu_freq = int(gpu_freq[0])

gpu_mem = gpu_mem.decode("utf-8")
gpu_mem = gpu_mem.split("\n")

gpu_mem = int(gpu_mem[0])

gpu_percent = gpu_percent.decode("utf-8")
gpu_percent = gpu_percent.split("\n")
gpu_percent = gpu_percent[0].split(",")
gpu_percent = gpu_percent[0].split("=")

gpu_percent = int(gpu_percent[1])
```

- **Impressão dos Valores Coletados**

Após o tratamento, foram também mantidas as impressões das informações coletadas, para uma visualização mais rápida dos dados enquanto o código está sendo executado.

```
net_sts = round(net_sts,2)

################################## Printa os valores ##############################
print("\n#########################################################################")

print("\n#", computer.processor.name)
print("Uso da CPU:           ", cpu_percent, "%")
print("Freq. da CPU:         ", cpu_freq, "Mhz")
print("Temp. da CPU:         ", cpu_temp, "°C")

print("\n#", gpu)
print("Uso da GPU:           ", gpu_percent, "%")
print("Freq. da GPU:         ", gpu_freq, "Mhz")
print("Temp. da GPU:         ", gpu_temp, "°C")
print("Uso da Mem. da GPU:   ", gpu_mem, "MB")

print("\n#", ram)
print("RAM Livre:            ", mem_free,"MB")
print("RAM Usada:            ", mem_used,"MB")

print("\n#", hd)
print("HDD Temp.:            ", hd_tmp, "°C")

print("\n#", ssd)
print("SSD Temp.:            ", ssd_tmp, "°C")

print("\n# Rede")
print("Uso de banda:         ", net_sts, "Kbps")

#print("\n# Bateria", chrg)
#print("Nível de Carga:      ", batt.percent, "%")

print("\n", data, hora)

#print("#########################################################################")
```

- **Envio para o Banco de Dados**

Por fim, os dados são enviados para três diferentes tabelas, que são divididas em Ocupação ou Uso, Velocidade ou Intensidade e Temperatura. Após os dados serem enviados o código da uma pausa pré definida para repetir o loop de coletas. Então, após a quantidade de loops escolhida acontecer, o servidor é fechado.

```
    insert_uso = "insert into Ocupação(CPU_Uso, GPU_Uso, GPU_MB, RAM_Livre, RAM_Usada)values(
        %s,%s,%s,%s,%s);"%(cpu_percent,gpu_percent,gpu_mem,mem_free,mem_used)

    insert_velocidade = "insert into Performance(CPU_Mhz, GPU_Mhz, Rede_Kbps)values(%s,%s,%s);
        "%(cpu_freq,gpu_freq,net_sts)

    insert_temperatura = "insert into Temperaturas(CPU_ºC, GPU_ºC, HDD_ºC, SSD_ºC)values(%s,%s
        ,%s,%s);"%(cpu_temp,gpu_temp,hd_tmp,ssd_tmp)

    cursor.execute(insert_uso)
    cursor.execute(insert_velocidade)
    cursor.execute(insert_temperatura)
    connection.commit()

    time.sleep(sleep)

cursor.close()
connection.close()
print("MySQL connection is closed")
```

D. Banco de Dados

As tabelas do banco de dados foram criadas graficamente através do programa dbeaver, os formatos foram escolhidos de acordo com o tipo de dado que seria posto em seu campo. A data e a hora foram armazenadas em forma de timestamp no mesmo campo, podendo assim serem um ponto de ligação preciso entre as tabelas.

```
CREATE TABLE `Ocupação` (
  `CPU_Uso` double DEFAULT NULL,
  `GPU_Uso` int(11) DEFAULT NULL,
  `GPU_MB` int(11) DEFAULT NULL,
  `RAM_Livre` double DEFAULT NULL,
  `RAM_Usada` double DEFAULT NULL,
  `Data_Hora` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

```
CREATE TABLE `Performance` (
  `CPU_Mhz` double DEFAULT NULL,
  `GPU_Mhz` int(11) DEFAULT NULL,
  `Rede_Kbps` double DEFAULT NULL,
  `Data_Hora` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

```
CREATE TABLE `Temperaturas` (
  `CPU_ºC` int(11) DEFAULT NULL,
  `GPU_ºC` int(11) DEFAULT NULL,
  `HDD_ºC` int(11) DEFAULT NULL,
  `SSD_ºC` int(11) DEFAULT NULL,
  `Data_Hora` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

III. RESULTADOS

A. Dados Coletados

A seguir será apresentada uma pequena amostra dos dados coletados, onde foi escolhido um período no qual o computador estava em stand by e começou a ser usado. Pode-se notar que as medições se iniciam na linha 301, e na linha 321 temos um claro indício de que o uso começou a mudar.

Na primeira figura temos as ocupações de CPU, GPU e RAM. Nota-se que no início todas as medições indicam o baixo uso, principalmente no caso da RAM que apresenta um uso de entorno de 1,5GB. Na linha 321 temos um pico de uso da CPU, seguido do aumento em todos as outros campos a partir daí.

	CPU_Uso	GPU_Uso	GPU_MB	RAM_Livre	RAM_Usada	Data_Hora
300	10.2	14	374	16,117.03	4,044.28	2020-09-18 15:30:08
301	1.7	22	311	19,038.24	1,332.29	2020-09-20 17:07:03
302	1.7	20	311	18,860.95	1,505.88	2020-09-20 17:08:07
303	2	17	312	18,833.52	1,533.08	2020-09-20 17:09:10
304	0.9	23	312	18,805.06	1,530.7	2020-09-20 17:10:13
305	2.2	21	312	18,804.39	1,531.42	2020-09-20 17:11:17
306	1.5	18	312	18,800.04	1,535.8	2020-09-20 17:12:20
307	0.9	21	312	18,806.29	1,529.48	2020-09-20 17:13:23
308	0.9	18	312	18,802.67	1,533.08	2020-09-20 17:14:26
309	1.4	19	311	18,801.02	1,534.72	2020-09-20 17:15:30
310	0.8	17	312	18,800.04	1,535.68	2020-09-20 17:16:33
311	2.8	31	312	18,798.63	1,537.11	2020-09-20 17:17:36
312	1.9	20	312	18,799.62	1,536.11	2020-09-20 17:18:40
313	0.6	18	312	18,800.5	1,535.23	2020-09-20 17:19:43
314	0.4	3	311	18,796.38	1,539.35	2020-09-20 17:20:46
315	0.6	3	311	18,798.24	1,537.45	2020-09-20 17:21:50
316	3.5	19	319	18,795.62	1,540.09	2020-09-20 17:22:53
317	1.9	20	319	18,794.84	1,540.84	2020-09-20 17:23:56
318	1.6	16	319	18,791.95	1,543.65	2020-09-20 17:25:00
319	0.8	19	319	18,787.89	1,547.72	2020-09-20 17:26:03
320	0.9	19	319	18,791.62	1,543.98	2020-09-20 17:27:06
321	22.7	18	391	17,481.67	2,665.48	2020-09-20 17:28:11
322	2.4	23	370	16,759.94	3,382.21	2020-09-20 17:29:14
323	14.6	18	414	16,163.21	3,948.45	2020-09-20 17:30:18
324	5.8	22	380	16,244.04	3,894.75	2020-09-20 17:31:21
325	1	9	393	16,118.77	4,012.6	2020-09-20 17:32:24
326	9.4	27	415	16,033.29	4,066.32	2020-09-20 17:33:28
327	4.1	29	402	16,013.23	4,085.36	2020-09-20 17:34:31
328	8.5	34	409	15,950.89	4,131.72	2020-09-20 17:35:34
329	6.1	28	405	15,966.19	4,119.08	2020-09-20 17:36:38
330	4	18	411	15,978.39	4,120.38	2020-09-20 17:37:41

Seguindo a lógica da figura anterior, aqui também temos um pico na frequência da CPU e após isto um aumento na frequência da GPU, além de demonstrar também o início do uso da rede.

	CPU_Mhz	GPU_Mhz	Rede_Kbps	Data_Hora
300	2,839.77	360	0.53	2020-09-18 15:30:08
301	900.52	375	0.1	2020-09-20 17:07:03
302	996.95	375	0	2020-09-20 17:08:07
303	1,849.31	375	0	2020-09-20 17:09:10
304	1,058.83	375	0	2020-09-20 17:10:13
305	950.06	375	0	2020-09-20 17:11:17
306	1,447.28	375	0	2020-09-20 17:12:20
307	1,073.86	405	0	2020-09-20 17:13:23
308	997.47	390	0	2020-09-20 17:14:26
309	1,495.96	360	0	2020-09-20 17:15:30
310	1,950.08	360	0	2020-09-20 17:16:33
311	944.28	750	0	2020-09-20 17:17:36
312	1,012.25	390	0	2020-09-20 17:18:40
313	1,080.5	360	0	2020-09-20 17:19:43
314	1,100.23	300	0	2020-09-20 17:20:46
315	1,557.87	300	0	2020-09-20 17:21:50
316	2,476.4	675	0	2020-09-20 17:22:53
317	1,001.2	360	0	2020-09-20 17:23:56
318	1,549.27	360	0	2020-09-20 17:25:00
319	1,479.23	360	0	2020-09-20 17:26:03
320	1,033.08	360	0	2020-09-20 17:27:06
321	3,045.86	690	226.31	2020-09-20 17:28:11
322	2,364.39	300	3.74	2020-09-20 17:29:14
323	1,297.91	705	9.1	2020-09-20 17:30:18
324	1,694.75	600	0	2020-09-20 17:31:21
325	2,348.41	420	0	2020-09-20 17:32:24
326	1,016.85	915	62.97	2020-09-20 17:33:28
327	2,177.43	300	1.56	2020-09-20 17:34:31
328	1,020.82	945	0.16	2020-09-20 17:35:34
329	900.04	375	11.53	2020-09-20 17:36:38
330	1,588.75	300	0.26	2020-09-20 17:37:41

Mostrando os efeitos do aumento do uso do sistema, essa última figura mostra os dados de temperatura. Na mesma linha 321 mostra um pico de temperatura na CPU e também o início do aumento da temperatura da GPU.

	CPU_°C	GPU_°C	HDD_°C	SSD_°C	Data_Hora
300	61	50	47	45	2020-09-18 15:30:08
301	34	31	21	45	2020-09-20 17:07:03
302	35	33	22	45	2020-09-20 17:08:07
303	34	34	24	45	2020-09-20 17:09:10
304	35	35	25	45	2020-09-20 17:10:13
305	38	35	26	45	2020-09-20 17:11:17
306	39	36	26	45	2020-09-20 17:12:20
307	37	37	27	45	2020-09-20 17:13:23
308	40	37	28	45	2020-09-20 17:14:26
309	40	38	29	45	2020-09-20 17:15:30
310	39	38	29	45	2020-09-20 17:16:33
311	44	39	30	29	2020-09-20 17:17:36
312	42	40	31	45	2020-09-20 17:18:40
313	48	40	31	45	2020-09-20 17:19:43
314	42	40	32	45	2020-09-20 17:20:46
315	42	40	32	45	2020-09-20 17:21:50
316	42	40	33	45	2020-09-20 17:22:53
317	43	42	33	45	2020-09-20 17:23:56
318	42	42	34	45	2020-09-20 17:25:00
319	47	43	34	45	2020-09-20 17:26:03
320	44	44	34	45	2020-09-20 17:27:06
321	84	49	35	38	2020-09-20 17:28:11
322	52	50	35	45	2020-09-20 17:29:14
323	50	47	35	45	2020-09-20 17:30:18
324	45	44	35	45	2020-09-20 17:31:21
325	55	46	35	45	2020-09-20 17:32:24
326	50	48	36	45	2020-09-20 17:33:28
327	51	49	36	45	2020-09-20 17:34:31
328	52	50	36	45	2020-09-20 17:35:34
329	52	50	37	45	2020-09-20 17:36:38
330	54	45	37	45	2020-09-20 17:37:41

B. Consultas ao Banco de Dados

Neste bloco serão mostradas algumas consultas feitas no banco de dados.

```
select Ocupação.CPU_Uso, Performance.CPU_Mhz, Temperaturas.CPU_ºC, Temperaturas.Data_Hora from Ocupação
inner join  Performance on Performance.Data_Hora = Ocupação.Data_Hora
inner join Temperaturas on Temperaturas.Data_Hora = Performance.Data_Hora group by Data_Hora order by CPU_Mhz desc;
```

Ocupação(+)

select Ocupação.CPU_Uso, Performance.CPU_Mhz, Enter a SQL expression to filter results (use Ctrl+Space)

	CPU_Uso	CPU_Mhz	CPU_ºC	Data_Hora
1	9.5	4,377.05	89	2020-09-21 17:33:35
2	9.2	4,374.24	89	2020-09-21 17:35:41
3	9.1	4,374.06	89	2020-09-21 17:36:44
4	9	4,355.33	91	2020-09-21 17:37:48
5	9.1	4,350.99	86	2020-09-21 17:32:31
6	9.1	4,325.48	94	2020-09-21 18:17:54
7	8.8	4,317.59	88	2020-09-21 17:40:57
8	9.3	4,317.29	89	2020-09-21 17:44:07
9	9.9	4,315.77	87	2020-09-21 17:34:38
10	8.9	4,315.09	90	2020-09-21 18:10:32
11	9.3	4,315.04	89	2020-09-21 17:42:01
12	9.2	4,312.49	89	2020-09-21 17:39:54
13	9.1	4,309.74	88	2020-09-21 18:11:35
14	9	4,307.51	90	2020-09-21 18:14:45
15	9	4,302.34	89	2020-09-21 18:16:51
16	8.9	4,300.16	92	2020-09-21 18:23:10
17	9.1	4,291.55	93	2020-09-21 18:13:41
18	10.5	4,290.1	80	2020-09-21 17:50:27
19	9	4,288.96	94	2020-09-21 18:25:17
20	8.8	4,284.56	90	2020-09-21 18:22:07
21	9.4	4,283.65	86	2020-09-21 18:12:38
22	11.2	4,283.03	88	2020-09-21 20:39:24
23	11.4	4,280.59	90	2020-09-21 20:36:14
24	10.2	4,272.8	88	2020-09-21 17:31:28

Na primeira consulta foi buscado o momento em que o CPU teve sua maior frequência registrada. Pode-se notar que todas as medições mostradas na imagem foram feitas no mesmo período de coleta de dados, neste momento uma aplicação em single core estava sendo executada, o que explica tanto a alta frequência quanto o baixo uso, pois em single core o processador tem maiores limites de frequência.

A próxima imagem mostra a consulta do maior uso de RAM registrado. O período é o mesmo da maior frequência, mas neste momento em vez de um processo sendo executado haviam 3 ou 4 threads em uso máximo. No ponto mais alto de uso de RAM notamos uma baixa no uso e da frequência do CPU acompanhado do aquecimento do SSD, o que provavelmente indica que foi iniciado o uso do swap, causando lentidão no sistema.

Na linha 9, onde o uso de RAM já havia se equilibrado, notamos também o pico mais alto no uso da CPU, e também a volta da alta temperatura e frequência mais elevada.

```
select Ocupação.CPU_Uso, Ocupação.GPU_Uso, Ocupação.RAM_Usada, Ocupação.RAM_Livre,
Performance.CPU_Mhz, Performance.GPU_Mhz,
Temperaturas.CPU_ºC, Temperaturas.SSD_ºC, Temperaturas.Data_Hora
from Ocupação
inner join  Performance on Performance.Data_Hora = Ocupação.Data_Hora
inner join Temperaturas on Temperaturas.Data_Hora = Performance.Data_Hora group by Data_Hora order by RAM_Usada desc;
```

Ocupação(+)

Enter a SQL expression to filter results (use Ctrl+Space)

	CPU_Uso	GPU_Uso	RAM_Usada	RAM_Livre	CPU_Mhz	GPU_Mhz	CPU_ºC	SSD_ºC	Data_Hora
1	16.1	31	19,073.68	869.64	2,688.51	795	57	51	2020-09-21 23:10:58
2	21.9	21	18,920.72	1,083.21	2,150.05	750	58	52	2020-09-21 23:09:53
3	19	39	18,801.51	1,245.44	2,547.97	1,155	52	45	2020-09-21 23:17:19
4	20.6	32	18,798.25	1,249.18	1,662.72	1,170	51	45	2020-09-21 23:16:16
5	12.9	39	18,794.21	1,254.73	1,321.99	855	51	45	2020-09-21 23:13:06
6	18.2	32	18,789.86	1,150.9	1,057.43	675	52	45	2020-09-21 23:15:13
7	21.9	20	18,788.45	1,153.52	2,249.28	810	53	47	2020-09-21 23:14:09
8	16.6	36	18,785.46	1,264.37	1,085.09	975	53	50	2020-09-21 23:12:02
9	45.2	34	17,456.89	2,227.81	3,723.23	900	95	48	2020-09-22 00:12:41
10	35.6	20	17,432.67	2,296.09	3,831.8	360	96	45	2020-09-22 00:11:36
11	36.6	21	17,404.99	2,324.47	3,800.61	360	91	47	2020-09-22 00:10:33
12	34.8	15	16,503.17	3,565.26	3,425.47	375	95	45	2020-09-22 00:41:12
13	36	19	16,406.29	3,667.98	3,436.17	525	95	45	2020-09-22 00:40:09
14	35.1	28	16,325.09	3,749.14	3,584.2	555	95	45	2020-09-22 00:39:05
15	37.6	17	16,312.98	3,761.57	3,497.53	585	96	45	2020-09-22 00:38:02
16	38.9	20	16,294.39	3,775.32	3,838.3	495	94	45	2020-09-22 00:30:38
17	36.6	20	16,291.61	3,782.96	3,701.55	525	94	45	2020-09-22 00:36:59
18	36.8	18	16,277.66	3,796.91	3,687.39	885	95	47	2020-09-22 00:35:55
19	35.9	15	16,273.74	3,800.85	3,699.3	720	95	45	2020-09-22 00:34:52
20	36.1	26	16,260.25	3,814.34	3,680.74	690	95	45	2020-09-22 00:33:48
21	38.5	23	16,229.05	3,845.44	3,753.9	465	95	45	2020-09-22 00:31:42
22	35.5	21	16,216.53	3,858.04	3,765.09	360	95	45	2020-09-22 00:32:45
23	37.3	20	16,172.03	3,902.76	3,797.89	690	95	47	2020-09-22 00:29:35
24	37.6	11	16,151.95	3,922.97	3,782.2	855	95	45	2020-09-22 00:27:28
25	35.8	22	16,151.59	3,923.21	3,721.03	405	97	45	2020-09-22 00:28:31

Para o monitoramento da temperatura o componente escolhido foi a CPU, por ter a maior variação e maior temperatura detectada, sendo seu maior pico de temperatura os 100ºC, que é a temperatura limite deste processador. Mais uma vez temos medições do período do dia 21 de setembro, no qual processos pesados estavam em execução.

Alguns dados podem ser observados nestes picos de temperatura. Primeiro é que os horários mais frequentes eram em torno das 17h, que foi o horário de início da execução dos processos, que coincidem também com frequências mais baixas que as medidas mais tarde. A explicação para isso é que, um tempo após a execução dos processos, foi feito o undervolting no processador, o que além de fazer com que as temperaturas se mantiveram mais baixas permitiu que o CPU pudesse se manter em frequências mais altas sem sobreaquecer.

```
select Ocupação.CPU_Uso, Ocupação.RAM_Usada, Ocupação.RAM_Livre, Performance.CPU_Mhz,
Temperaturas.CPU_ºC, Temperaturas.GPU_ºC, Temperaturas.HDD_ºC, Temperaturas.SSD_ºC, Temperaturas.Data_Hora
from Ocupação
inner join  Performance on Performance.Data_Hora = Ocupação.Data_Hora
inner join Temperaturas on Temperaturas.Data_Hora = Performance.Data_Hora group by Data_Hora order by CPU_ºC desc;
```

Ocupação(+)

select Ocupação.CPU_Uso, Ocupação.RAM_Usada, Oc Enter a SQL expression to filter results (use Ctrl+Space)

	CPU_Uso	RAM_Usada	RAM_Livre	CPU_Mhz	CPU_°C	GPU_°C	HDD_°C	SSD_°C	Data_Hora
1	10.4	9,187.39	10,958.53	4,180.35	100	49	40	45	2020-09-21 17:28:18
2	21.3	7,485.79	12,151.2	3,541.02	100	66	42	57	2020-09-16 23:18:38
3	19	5,967.04	14,015.53	4,015.48	100	58	40	45	2020-09-18 03:14:24
4	21.8	12,731.94	7,290.34	4,018.01	100	56	39	45	2020-09-21 00:43:33
5	25.4	6,771.67	12,926.85	3,858.33	100	60	39	45	2020-09-18 02:51:07
6	24.9	6,711.26	12,994.95	3,843.38	100	62	40	45	2020-09-18 02:56:25
7	23.1	6,720.68	12,979.84	3,683.81	100	60	39	48	2020-09-18 02:46:53
8	8.9	7,432.69	12,717.29	4,181.6	100	46	42	40	2020-09-21 17:22:10
9	23	12,669.58	7,362.7	3,781.25	100	55	39	45	2020-09-21 00:31:54
10	9.7	9,186.96	10,959.03	4,172.89	99	48	40	45	2020-09-21 17:29:22
11	10.1	7,611.42	12,534.39	4,213.6	99	51	41	45	2020-09-21 17:27:15
12	14.8	6,678.33	13,459.18	4,156.01	98	52	43	45	2020-09-21 17:19:00
13	27.5	7,498.69	12,139.69	3,592.37	97	66	42	57	2020-09-16 23:27:06
14	35.8	16,151.59	3,923.21	3,721.03	97	60	40	45	2020-09-22 00:28:31
15	35.6	15,092.45	4,984.02	3,790.82	97	59	40	45	2020-09-22 00:16:54
16	37.6	16,312.98	3,761.57	3,497.53	96	61	40	45	2020-09-22 00:38:02
17	24.5	7,521.81	12,118.34	3,963.51	96	67	42	58	2020-09-16 23:28:09
18	37.1	14,336.5	5,642.92	3,715.96	96	63	40	45	2020-09-22 00:52:49
19	19.1	10,486.08	9,486.68	4,052.22	96	64	35	45	2020-09-21 19:46:45
20	36	15,850.55	4,225.92	3,707.52	96	60	40	45	2020-09-22 00:24:18
21	38.9	14,381.76	5,599.19	3,727.07	96	63	39	45	2020-09-22 00:58:07
22	38.7	14,397.21	5,580.18	3,690.57	96	65	39	49	2020-09-22 01:03:24
23	35	15,271.74	4,804.76	3,660.53	96	59	40	45	2020-09-22 00:20:04
24	17.2	10,577.06	9,393.53	4,091.34	96	65	35	45	2020-09-21 19:57:16
25	19.8	10,737.26	9,278.41	4,073.17	96	65	36	45	2020-09-21 20:02:32

Na última consulta feita foi analisada às medições da GPU sendo ordenadas através da maior frequência capturada. O que podemos notar é uma enorme diferença entre a GPU e a CPU no quesito de temperatura, pois mesmo em maiores frequências, maiores quantidades de memória de vídeo em uso e maiores usos da GPU ela se mantém em uma temperatura próxima a de standby da CPU.

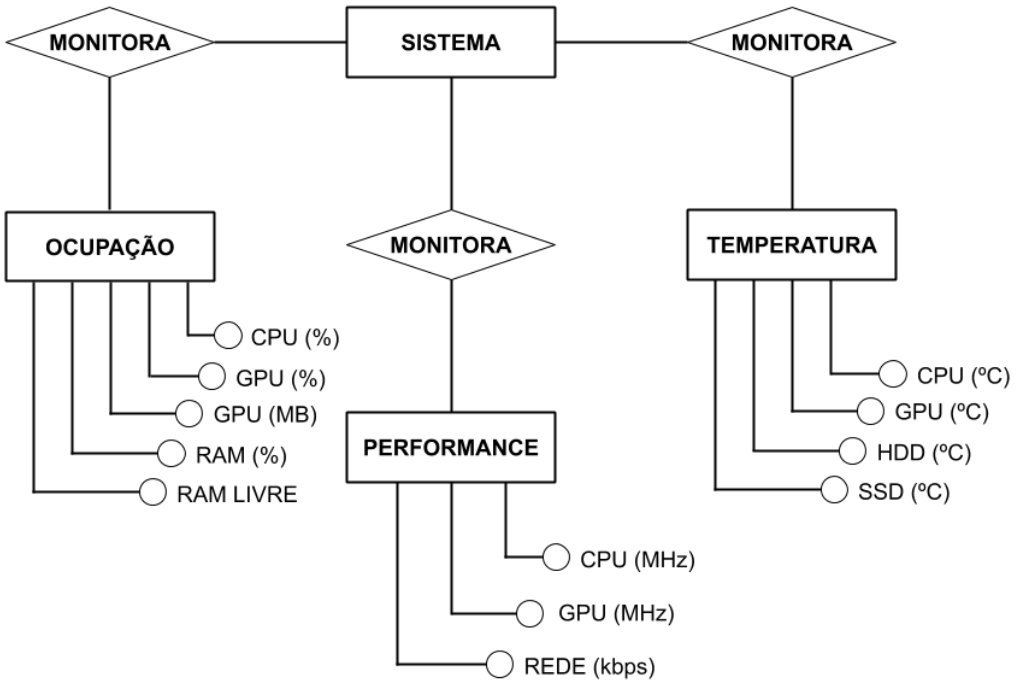
```
select Ocupação.GPU_Uso, Ocupação.GPU_MB, Performance.GPU_Mhz,
Temperaturas.GPU_ºC, Temperaturas.Data_Hora
from Ocupação
inner join  Performance on Performance.Data_Hora = Ocupação.Data_Hora
inner join Temperaturas on Temperaturas.Data_Hora = Performance.Data_Hora group by Data_Hora order by GPU_Mhz desc;
```

Ocupação(+)

Enter a SQL expression to filter results (use Ctrl+Space)

	GPU_Uso	GPU_MB	GPU_Mhz	GPU_ºC	Data_Hora
1	46	384	1,845	55	2020-09-18 14:58:03
2	37	723	1,845	56	2020-09-16 20:13:49
3	40	708	1,830	62	2020-09-16 20:25:25
4	39	709	1,830	59	2020-09-16 20:14:53
5	40	585	1,830	59	2020-09-16 20:38:04
6	38	707	1,830	61	2020-09-16 20:15:56
7	37	710	1,830	61	2020-09-16 20:28:35
8	40	708	1,830	63	2020-09-16 20:29:38
9	38	707	1,830	64	2020-09-16 20:30:41
10	41	712	1,830	65	2020-09-16 20:31:45
11	39	708	1,830	62	2020-09-16 20:22:16
12	36	708	1,830	63	2020-09-16 20:24:22
13	38	710	1,815	61	2020-09-16 20:27:32
14	33	370	1,800	52	2020-09-18 14:56:59
15	38	700	1,755	61	2020-09-16 20:19:06
16	38	700	1,755	61	2020-09-16 20:21:12
17	38	715	1,740	63	2020-09-16 20:32:48
18	36	854	1,695	68	2020-09-21 20:18:20
19	38	777	1,665	67	2020-09-21 20:09:55
20	38	724	1,650	63	2020-09-16 20:34:54

C. Diagrama E-R



No DER podemos ver a forma na qual foram organizadas as tabelas do banco de dados. De início as tabelas seriam separadas por hardware, mas isso geraria várias tabelas com apenas uma ou duas colunas. Por este motivo a forma escolhida para separar os dados foi baseada no tipo de informação e em que este dado pode influenciar, ou seja, cada tabela irá informar um estado do computador.

A tabela Ocupação irá informar o quanto o de processamento e memória está sendo usado, podendo evitar que mais processos sejam iniciados com o computador estando com um alto nível de uso.

A tabela Performance tem informações da velocidade da CPU, GPU e da Rede. Isto pode ser usado de comparativo com outras informações como temperatura e uso, pois, por exemplo, apenas pelo fato da CPU estar operando em alta frequência não quer dizer que mais processos não possam ser iniciados, porque ainda podem ter núcleos do processador sem uso nenhum. Mas também uma alta frequência, em alguns casos, pode causar um aquecimento que poderá ser monitorado na última tabela.

Na tabela temperatura, além da CPU, temos temperatura de Discos (HD e SSD) e GPU, com essas informações o uso do computador pode ser ajustado para uma melhor longevidade do hardware. Operar sempre em altas temperaturas diminui a vida útil dos componentes, por isso é importante saber os momentos em que elas ocorrem para poder detectar suas origens resolvê-las.

IV. CONCLUSÃO

Esta etapa do projeto foi concluída obtendo sucesso em seu objetivo. Os dados coletados são adequadamente armazenados no banco de dados de forma organizada e segura, possibilitando uma fácil análise de comportamento do sistema monitorado.

Observando os dados coletados pôde-se descobrir alguns comportamentos do sistema. Foi notável o desempenho positivo do processo de undervolting na CPU, que manteve as temperaturas mais baixas e ainda possibilitou um maior desempenho, através do impedimento do clock throttling, se mostrando algo indispensável para notebooks ou sistemas em geral com problemas de altas temperaturas.

Tendo tais resultados importantes após rápidos monitoramentos em um computador de uso residencial, prova-se que um sistema de monitoramento deste tipo pode ser de extrema importância em dispositivos de usos mais intensos e contínuos, onde sobreaquecimentos podem causar lentidão no sistema e até danificar o hardware.