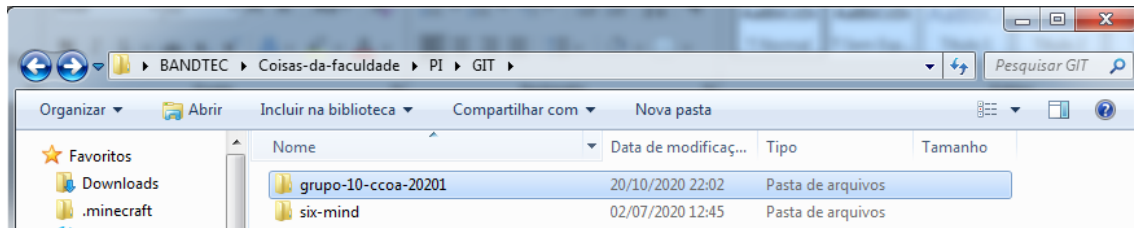
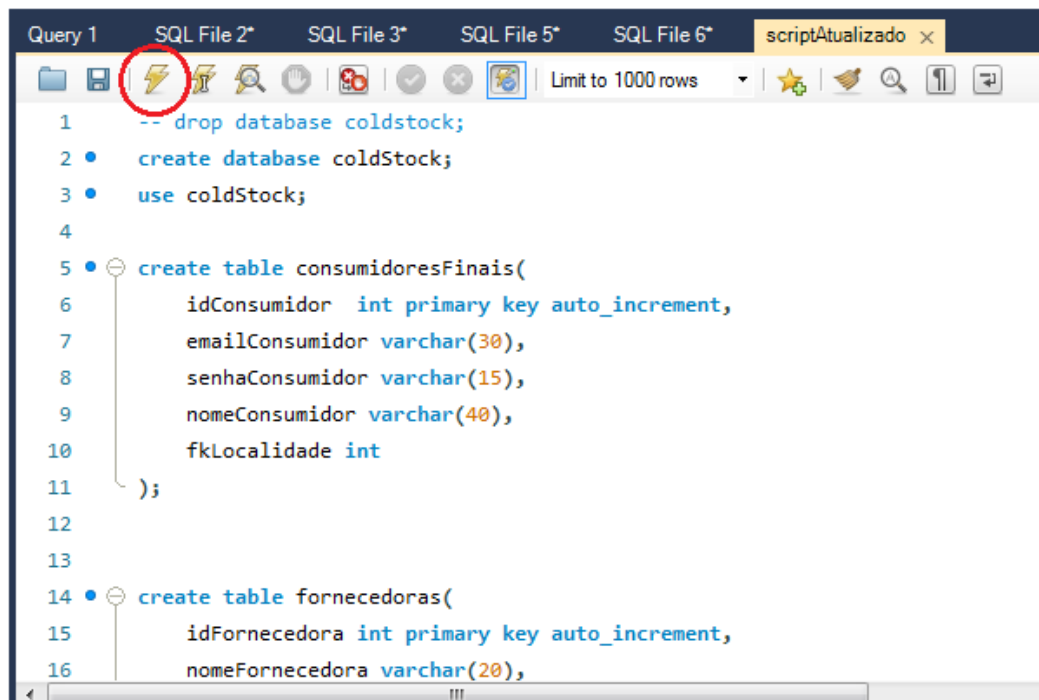
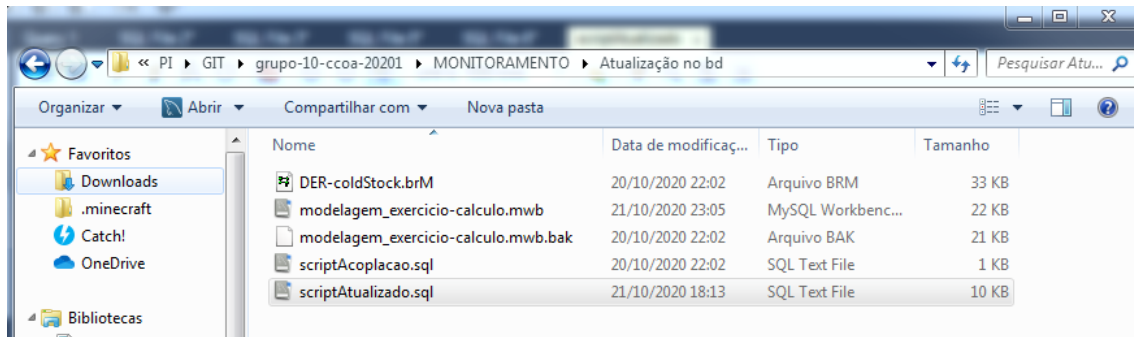


Relatório das API's

Primeiramente vamos ao GitHub e vamos clonar o repositório das nossas API:



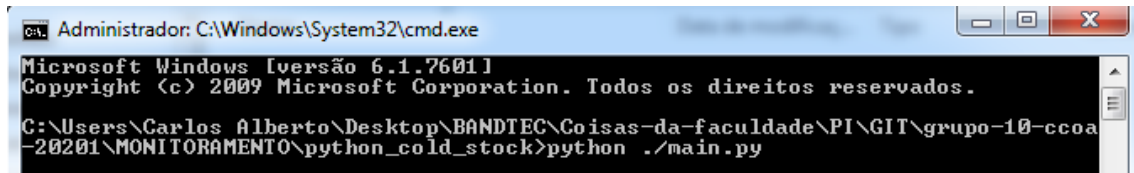
Depois disso precisamos criar as tabelas no nosso banco de dados para que possamos inserir e visualizar os dados que iremos gerar. Dentro do nosso repositório temos o script que cria todas as tabelas que precisamos então vamos até o repositório onde está nosso script e rodamos ele no Mysql Workbench:



Com todas as nossas tabelas já criadas vamos para a parte de rodar nossas API's. Nós estamos utilizando duas API's, uma utiliza os comandos psutil,

e a outra API é um web crawler que utiliza o OpenHardwareMonitor para capturar os dados da nossa máquina.

Api pythonycs: Para nós rodarmos nossa api pythonycs basicamente entramos na pasta python_cold_stock com o nosso cmd e inserimos o comando: “python ./main.py”



```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [versão 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.

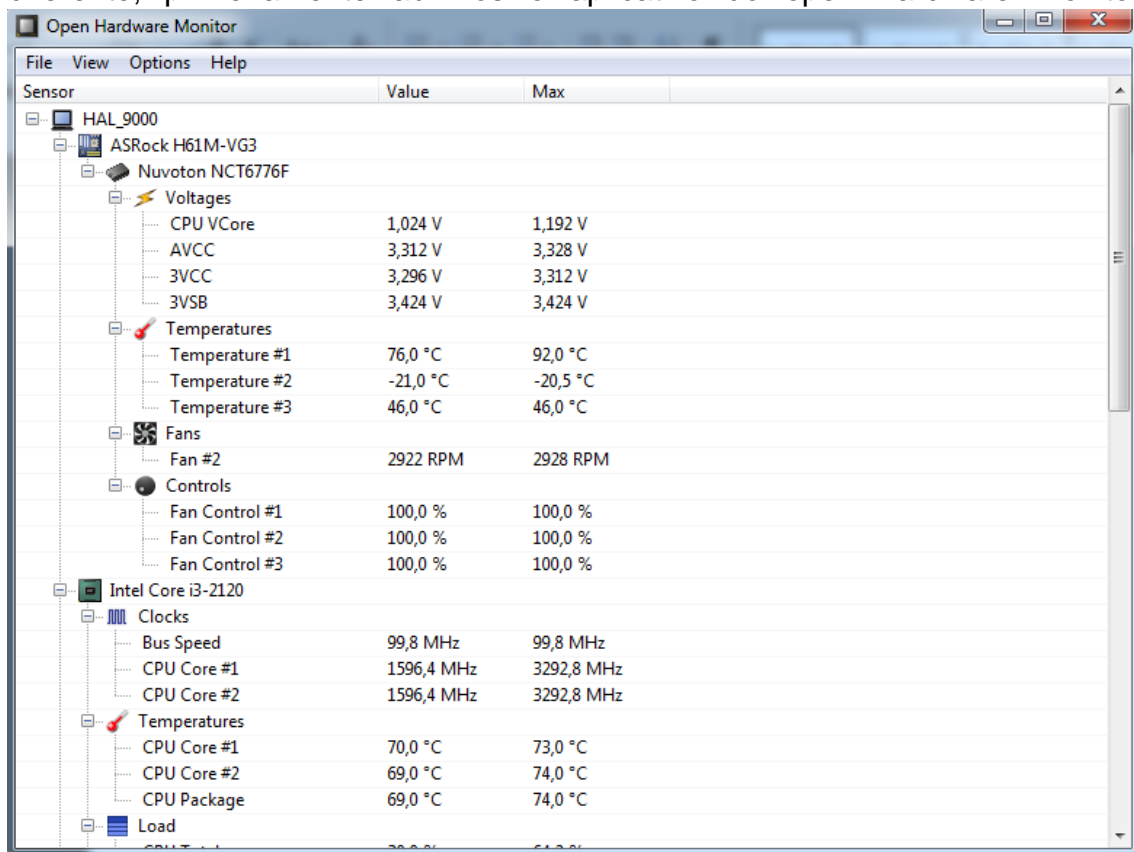
C:\Users\Carlos Alberto\Desktop\BANDTEC\Coisas-da-faculdade\PI\GIT\grupo-10-ccoa-20201\MONITORAMENTO\python_cold_stock>python ./main.py
```

Com esse comando a nossa API já faz os inserts dos dados capturados por comando da biblioteca psutil, esses dados são de memória, disco, conexão de dowload e upload

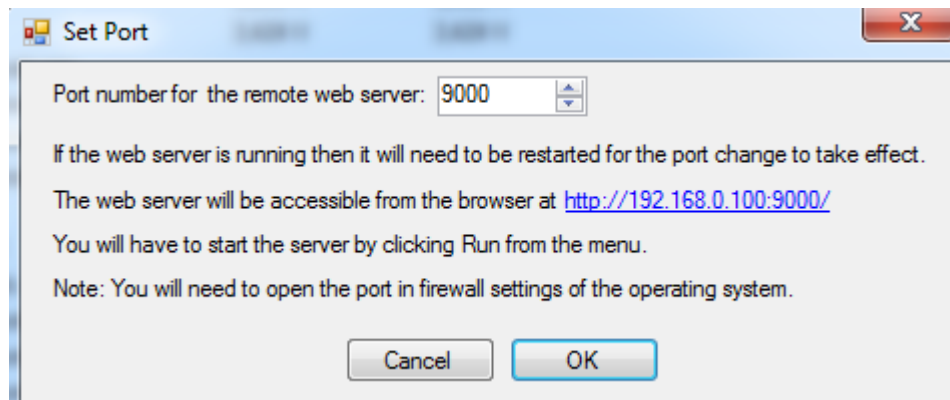
idRegistro	dataHora	valor	fkComponente	fkMaquina
729	2020-10-22 23:51:55	2.55	1	1
730	2020-10-22 23:51:55	4.4	2	1
731	2020-10-22 23:51:55	380.87	3	1
732	2020-10-22 23:51:55	11.39	4	1
733	2020-10-22 23:51:55	1.41	5	1

Nossa aplicação utiliza uma tabela modular, ou seja, não fizemos uma coluna para cada componente analisado e sim uma coluna com todos os valores captados e com um fkComponente indicando a qual componente aquele valor se refere.

API pythOHNs: Para rodar nossa API pythonhs temos um caminho diferente, primeiramente abrimos o aplicativo do open hardware monitor



No aplicativo vamos em Options -> Remote Web Server -> run, assim o aplicativo roda no nosso localhost e setamos para que usamos a porta 9000



Então vamos até o nosso navegador e entramos no endereço <http://192.168.0.100:9000/> e assim os dados são enviados ao nosso navegador e assim a API pode funcionar como um Web Crawler e capturar os dados do navegador em si

Sensor	Min	Value	Max
HAL_9000			
ASRock H61M-VG3			
Nuvoton NCT6776F			
Voltages			
CPU VCore	0,952 V	1,096 V	1,192 V
Voltage #2	0,184 V	0,192 V	0,192 V
AVCC	3,296 V	3,312 V	3,328 V
3VCC	3,280 V	3,296 V	3,312 V
Voltage #5	0,568 V	0,568 V	0,576 V
Voltage #6	1,704 V	1,712 V	1,720 V
Voltage #7	1,776 V	1,792 V	1,800 V
3VSB	3,408 V	3,424 V	3,424 V
Temperatures			
Temperature #1	46,5 °C	81,0 °C	93,0 °C
Temperature #2	-22,0 °C	-21,0 °C	-20,0 °C
Temperature #3	45,0 °C	46,0 °C	46,0 °C
Fans			
Fan #2	2896 RPM	2928 RPM	2986 RPM
Controls			
Fan Control #1	100,0 %	100,0 %	100,0 %
Fan Control #2	100,0 %	100,0 %	100,0 %
Fan Control #3	100,0 %	100,0 %	100,0 %
Intel Core i3-2120			
Clocks			

Após esse passo nós apenas precisamos entrar na pasta pythonHM_coldStock com o cmd e executar o comando: “python ./main.py”

```

C:\Users\Carlos Alberto\Desktop\BANDTEC\Coisas-da-faculdade\PI\GIT\grupo-10-ccoa-20201\MONITORAMENTO\pythonHM_coldStock>python ./main.py

```

```

C:\Users\Carlos Alberto\Desktop\BANDTEC\Coisas-da-faculdade\PI\GIT\grupo-10-ccoa-20201\MONITORAMENTO\pythonHM_coldStock>python ./main.py
Configurando BD
Conectando ao BD
<mysql.connector.connection.MySQLConnection object at 0x0074E5C8>
Iniciando o loop
Selecionando dados do server ID: 2
Retorno do BD: [<'CPU', 85, 1>, <'RAM', 74, 2>, <'Disco', 87, 3>]
Numero de registros: 3
[3.4, '4.4', 70.0]
Abaixo é nossa lista:
[['2020-10-23 00:09:16', 3.4, 2, 1], ['2020-10-23 00:09:16', '4.4', 2, 2], ['2020-10-23 00:09:16', 70.0, 2, 3]]
Aguarde ...
Selecionando dados do server ID: 2
Retorno do BD: [<'CPU', 85, 1>, <'RAM', 74, 2>, <'Disco', 87, 3>]
Numero de registros: 3
[1.7, '4.5', 67.67]
Abaixo é nossa lista:
[['2020-10-23 00:09:22', 1.7, 2, 1], ['2020-10-23 00:09:22', '4.5', 2, 2], ['2020-10-23 00:09:22', 67.67, 2, 3]]
Aguarde ...

```

A API pythohns capta dados de frequência de CPU, RAM e a temperatura do CPU, e faz insert desses dados no mesmo formato da outra API citada acima, então por isso utilizamos a mesma tabela no banco de dados e referenciamos os dados dos valores com campo fkComponente

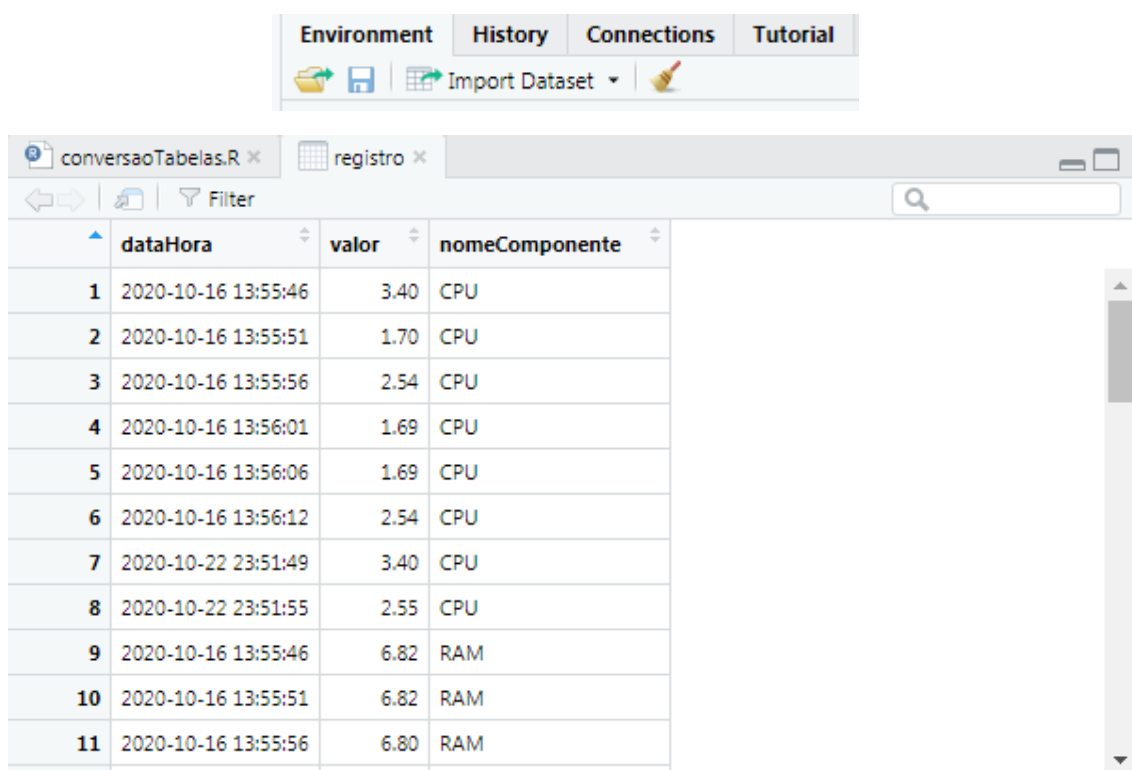
idRegistro	dataHora	valor	fkComponente	fkMaquina
735	2020-10-23 00:09:16	3.4	1	2
736	2020-10-23 00:09:16	4.4	2	2
737	2020-10-23 00:09:16	70	6	2

Nossas duas API's estão rodando e fazendo os inserts de forma correta, então agora vamos fazer a modelagem matemática dos dados utilizando a ferramenta R studio, então primeiramente vamos exportar a tabela registros para que possamos modelar esse dados e simular uma máquina servidor por exemplo

Importamos a tabela registro do nosso banco de dados no formato .csv utilizando o Table Data Export Wizard com o seguinte comando select:

```
"select dataHora, valor, nomeComponente from coldstock.registros  
INNER JOIN coldstock.componentes on fkComponente = idComponente  
where fkMaquina = 1 order by dataHora;"
```

Após exportarmos essa tabela que chamamos de "registro.csv" importamos ela para o R

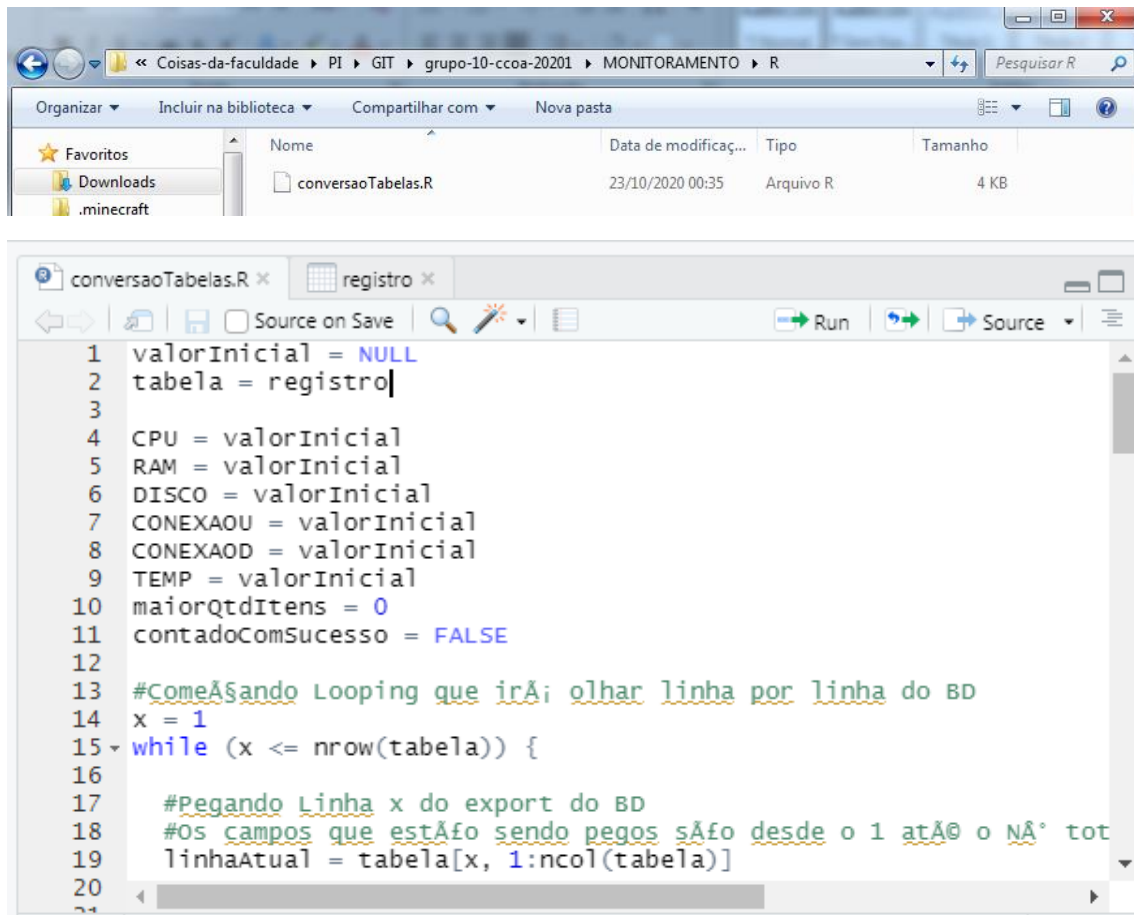


The screenshot shows the R Studio interface. At the top, there are tabs for 'Environment', 'History', 'Connections', and 'Tutorial'. Below these is a toolbar with icons for file operations and a dropdown menu labeled 'Import Dataset'. The main workspace area displays a table named 'registro' with the following data:

	dataHora	valor	nomeComponente
1	2020-10-16 13:55:46	3.40	CPU
2	2020-10-16 13:55:51	1.70	CPU
3	2020-10-16 13:55:56	2.54	CPU
4	2020-10-16 13:56:01	1.69	CPU
5	2020-10-16 13:56:06	1.69	CPU
6	2020-10-16 13:56:12	2.54	CPU
7	2020-10-22 23:51:49	3.40	CPU
8	2020-10-22 23:51:55	2.55	CPU
9	2020-10-16 13:55:46	6.82	RAM
10	2020-10-16 13:55:51	6.82	RAM
11	2020-10-16 13:55:56	6.80	RAM

Então voltamos ao diretório do nosso projeto e vamos atrás do script que criamos para alterar nossa tabela de forma com que possamos modelá-la de

forma mais simples. Já que fizemos uma tabela modular modificamos ela com um script em R para que cada componente tenha um coluna específica



Assim criando essa rodando esse script criamos uma tabela chamada “novaTabela” e a partir dela modelamos de forma simples e fácil os dados capturados

```

> novaTabela
  ID CPU RAM DISCO CONEXAOD CONEXAOU TEMP
1  1 3.40 6.82 379.91      8.92      1.45 70.67
2  2 1.70 6.82 379.91      8.92      1.46 66.00
3  3 2.54 6.80 379.91      8.92      1.46 64.67
4  4 1.69 6.80 379.91      8.94      1.46 64.00
5  5 1.69 6.80 379.91      8.94      1.46 62.67
6  6 2.54 6.81 379.91      8.94      1.46 61.67
7  7 3.40 4.40 380.87     11.39      1.41 70.67
8  8 2.55 4.40 380.87     11.39      1.41 68.67
> |

```

No exemplo abaixo vamos simular o CPU de um servidor multiplicando os valores capturado por 3, assim sendo um servidor 3 vezes mais potente que nossa máquina

```

> dadosServidor <- novaTabela$CPU*3
> dadosServidor
[1] 10.20  5.10  7.62  5.07  5.07  7.62 10.20  7.65
> |

```