

API PYTHOAMS

Na API Pythonics utilizamos o pacote **PSutil**, para capturar **dados de CPU, Memória e Disco**, funcionando muito bem em sistemas operacionais Linux e Windows. No entanto, o pacote PSutil tem limites em sistemas operacionais baseados em Windows 10, pois a BIOS é mantida em modo protegida, o que não permite capturar os **dados de temperatura e frequência (clock)**.

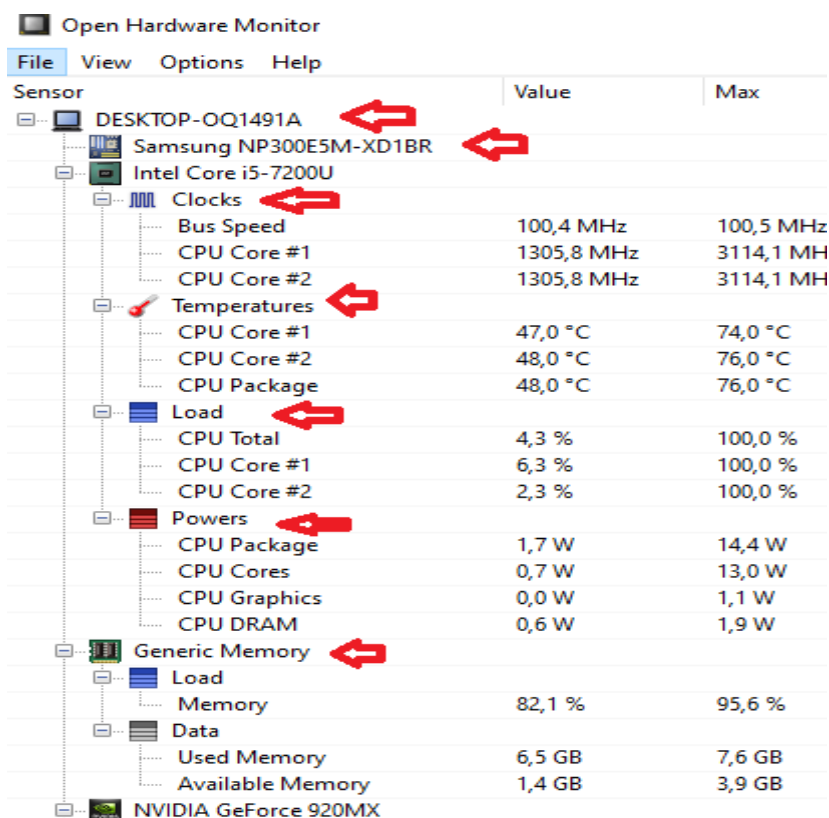
A Microsoft disponibiliza o módulo WMI (Windows Management Instrumentation) que fornece notificações e informações sobre o hardware. Neste caso, disponibiliza a Thermal Zone como sendo a temperatura da motherboard, que erroneamente tem sido adotada como temperatura de CPU.

Para resolver este problema, vamos utilizar o executável Open Hardware Monitor - OHM, é implementado em C#, possui dll nativa, capaz de capturar temperatura e frequência **diretamente da BIOS**, referindo-se a CPU, os seus "cores" e o clock. Funciona muito bem na versão para Windows e para Linux, está sobre licença free de uso.

O executável OHM possui uma funcionalidade que pode ser setada para rodar via browser, permitindo acessar conteúdo via APIs web baseadas em estruturas no formato Json.

O **JSON – JavaScript Object Notation**, é um padrão, com um formato, que permite a troca de dados entre programas, sendo amplamente utilizado na web. Cada linguagem fornece uma maneira de transformar dados no formato JSON em objetos nativos daquela linguagem e vice-versa. A nossa API pode capturar dados desse formato seguindo o padrão disponibilizado, o que diminui o esforço de desenvolvimento.

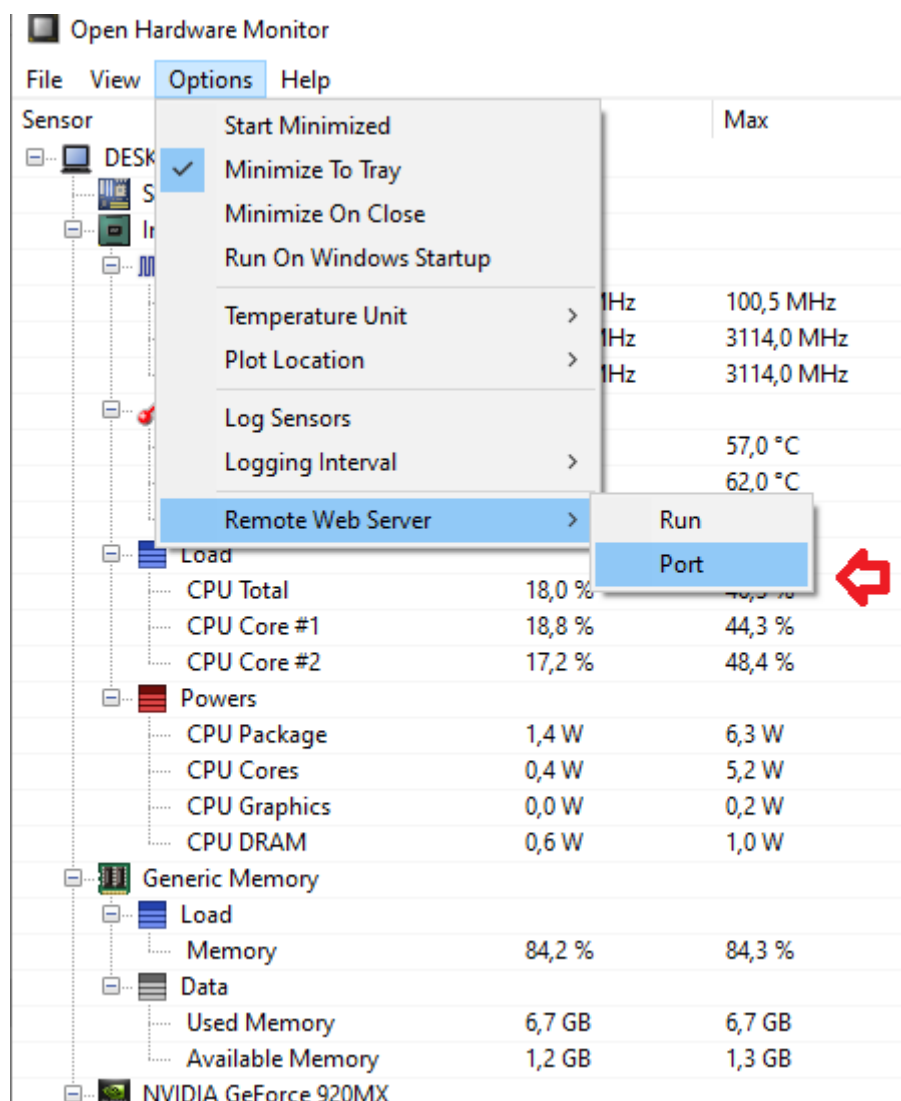
Vamos examinar o OHM:



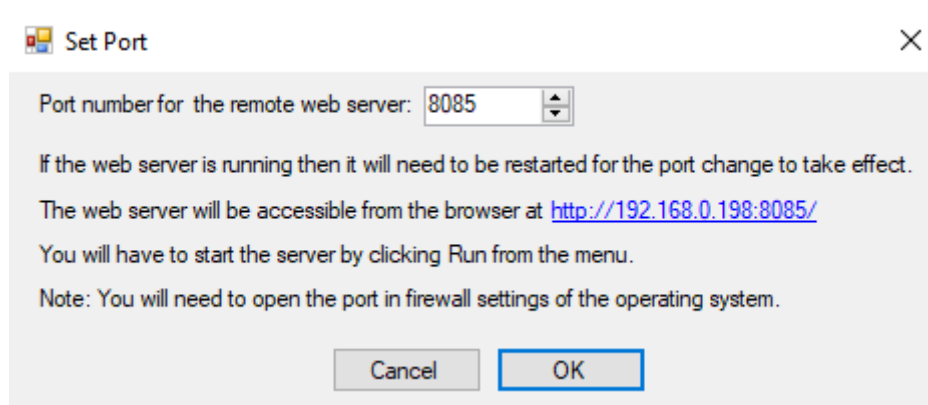
The screenshot shows the Open Hardware Monitor application window. The interface includes a menu bar (File, View, Options, Help) and a tree view on the left listing various hardware components. The main area displays a table of sensor data for each component. Red arrows point to specific sections: Desktop system info, Samsung SSD, Intel CPU, Clocks, Temperatures, Load, Powers, Generic Memory, and the NVIDIA GPU.

Sensor	Value	Max
DESKTOP-OQ1491A		
Samsung NP300E5M-XD1BR		
Intel Core i5-7200U		
Clocks		
Bus Speed	100,4 MHz	100,5 MHz
CPU Core #1	1305,8 MHz	3114,1 MHz
CPU Core #2	1305,8 MHz	3114,1 MHz
Temperatures		
CPU Core #1	47,0 °C	74,0 °C
CPU Core #2	48,0 °C	76,0 °C
CPU Package	48,0 °C	76,0 °C
Load		
CPU Total	4,3 %	100,0 %
CPU Core #1	6,3 %	100,0 %
CPU Core #2	2,3 %	100,0 %
Powers		
CPU Package	1,7 W	14,4 W
CPU Cores	0,7 W	13,0 W
CPU Graphics	0,0 W	1,1 W
CPU DRAM	0,6 W	1,9 W
Generic Memory		
Load		
Memory	82,1 %	95,6 %
Data		
Used Memory	6,5 GB	7,6 GB
Available Memory	1,4 GB	3,9 GB
NVIDIA GeForce 920MX		

API PYTHOAMS

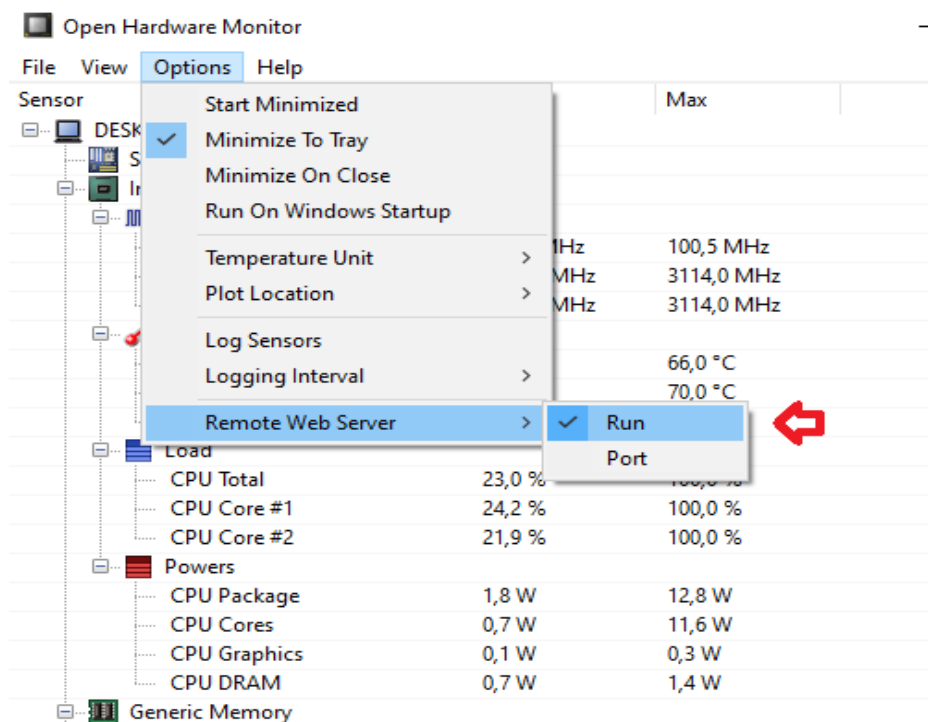


Observe que a ferramenta permite habilitar o servidor remoto web via porta 8085, no endereço padrão <http://192.168.0.198:8085>.

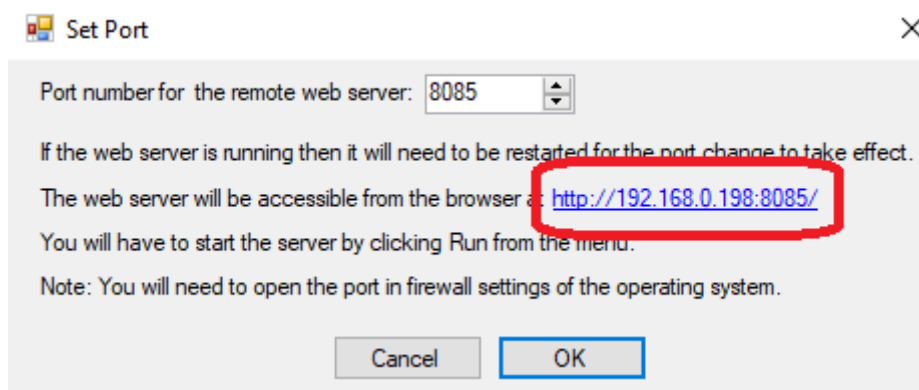


API PYTHOAMS

Setamos também para rodar via servidor web remoto

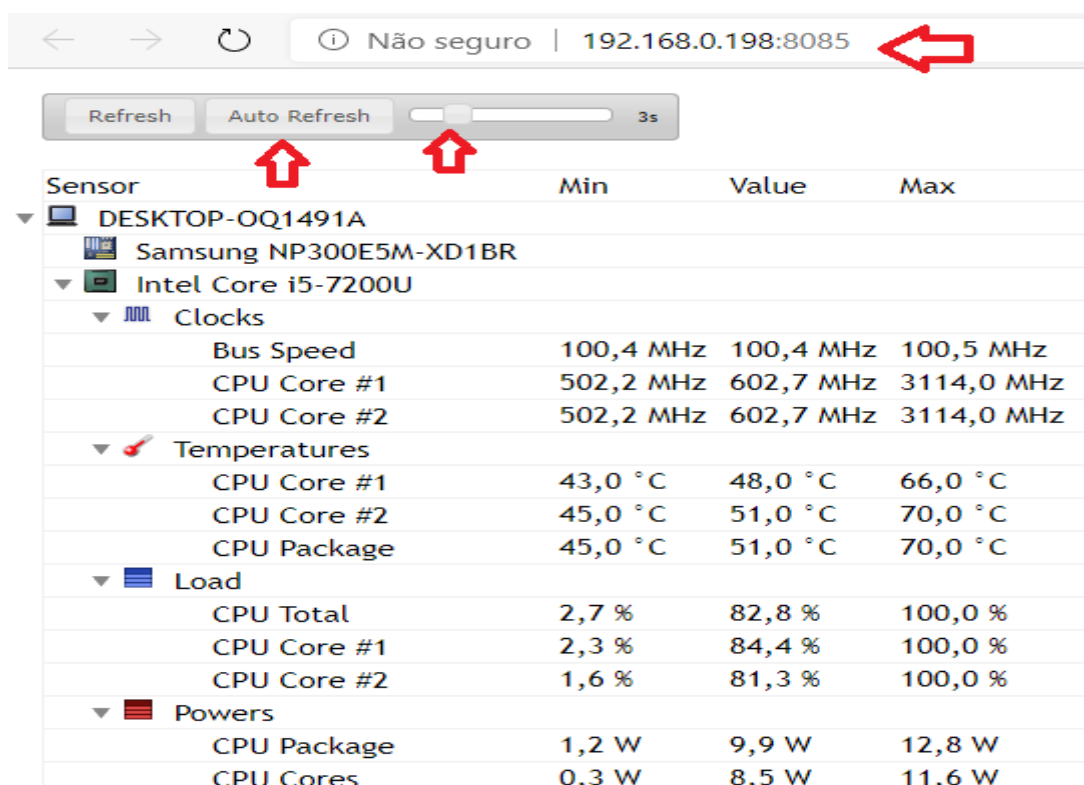


Depois disso é só clicar no endereço para que a ferramenta rod via browser



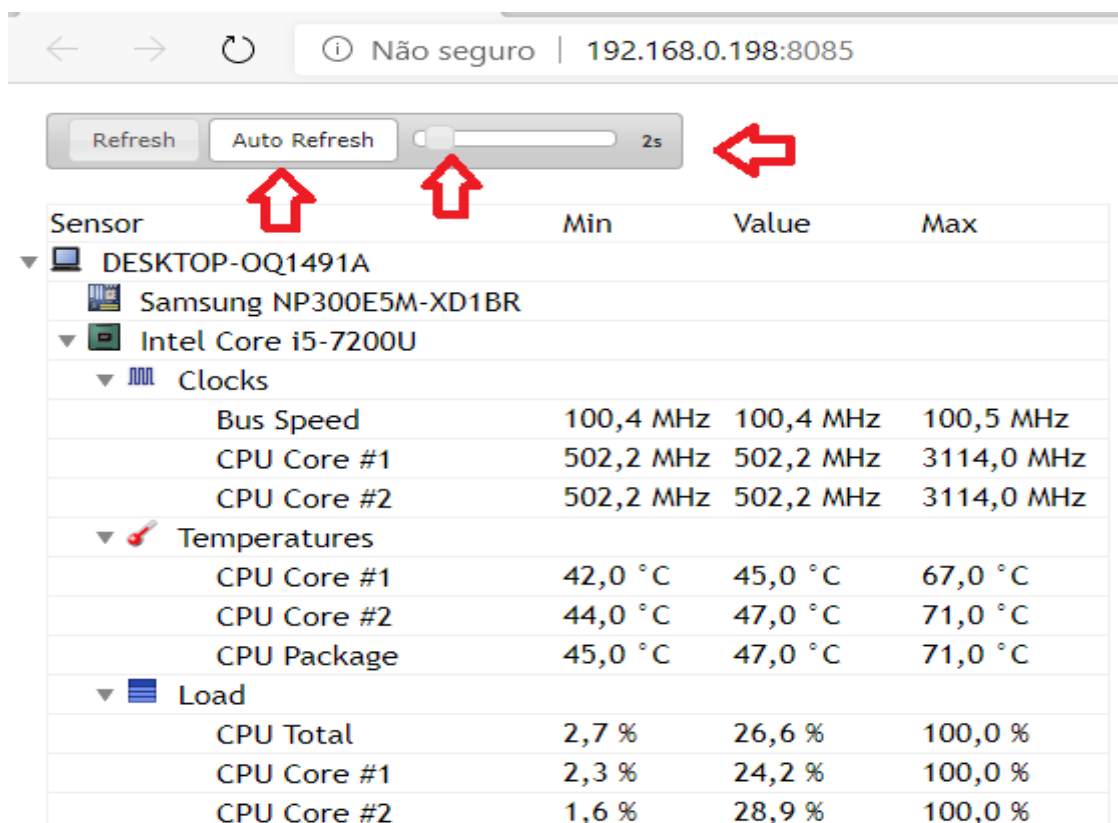
Vai abrir o ambiente remoto:

API PYTHOHMS



Sensor	Min	Value	Max
DESKTOP-OQ1491A			
Samsung NP300E5M-XD1BR			
Intel Core i5-7200U			
Clocks			
Bus Speed	100,4 MHz	100,4 MHz	100,5 MHz
CPU Core #1	502,2 MHz	602,7 MHz	3114,0 MHz
CPU Core #2	502,2 MHz	602,7 MHz	3114,0 MHz
Temperatures			
CPU Core #1	43,0 °C	48,0 °C	66,0 °C
CPU Core #2	45,0 °C	51,0 °C	70,0 °C
CPU Package	45,0 °C	51,0 °C	70,0 °C
Load			
CPU Total	2,7 %	82,8 %	100,0 %
CPU Core #1	2,3 %	84,4 %	100,0 %
CPU Core #2	1,6 %	81,3 %	100,0 %
Powers			
CPU Package	1,2 W	9,9 W	12,8 W
CPU Cores	0,3 W	8,5 W	11,6 W

Veja que está rodando no browser, clique em autorefresh e ajuste para 2s (segundos) para que o ambiente atualize as leituras.



Sensor	Min	Value	Max
DESKTOP-OQ1491A			
Samsung NP300E5M-XD1BR			
Intel Core i5-7200U			
Clocks			
Bus Speed	100,4 MHz	100,4 MHz	100,5 MHz
CPU Core #1	502,2 MHz	502,2 MHz	3114,0 MHz
CPU Core #2	502,2 MHz	502,2 MHz	3114,0 MHz
Temperatures			
CPU Core #1	42,0 °C	45,0 °C	67,0 °C
CPU Core #2	44,0 °C	47,0 °C	71,0 °C
CPU Package	45,0 °C	47,0 °C	71,0 °C
Load			
CPU Total	2,7 %	26,6 %	100,0 %
CPU Core #1	2,3 %	24,2 %	100,0 %
CPU Core #2	1,6 %	28,9 %	100,0 %

API PYTHOHMS

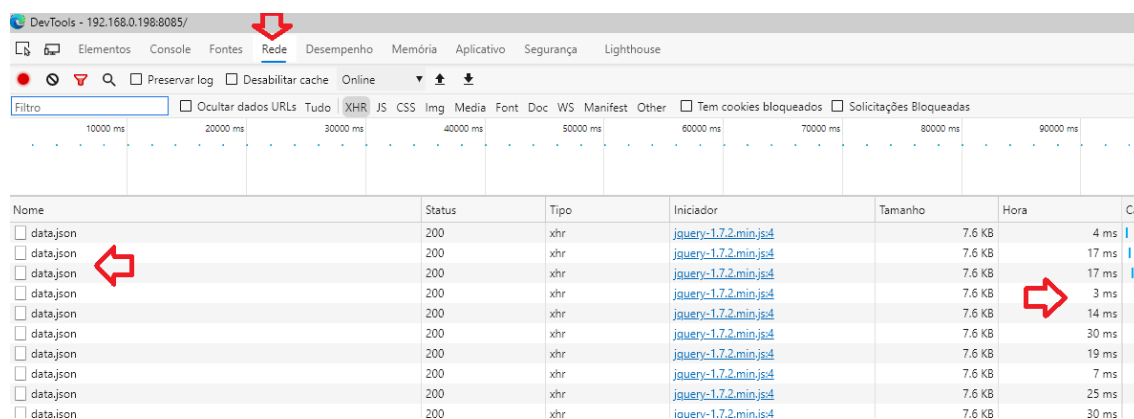
Agora vamos entender quais informações este OHM disponibiliza.

Clique com o botão direito do mouse na área vazia do browser com o OHM ativo rodando.

Clique em inspecionar

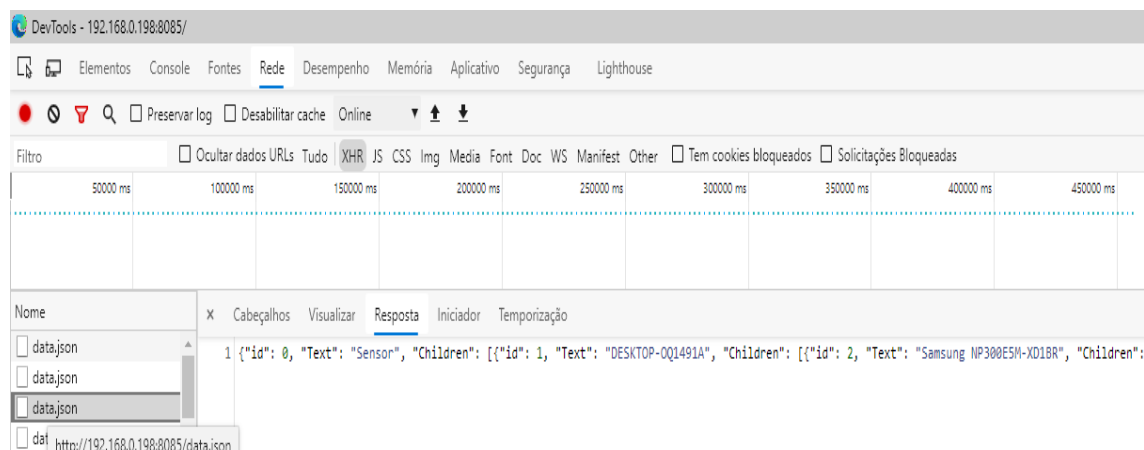
Vá em rede ou network

Dê um ctrl+R para atualizar



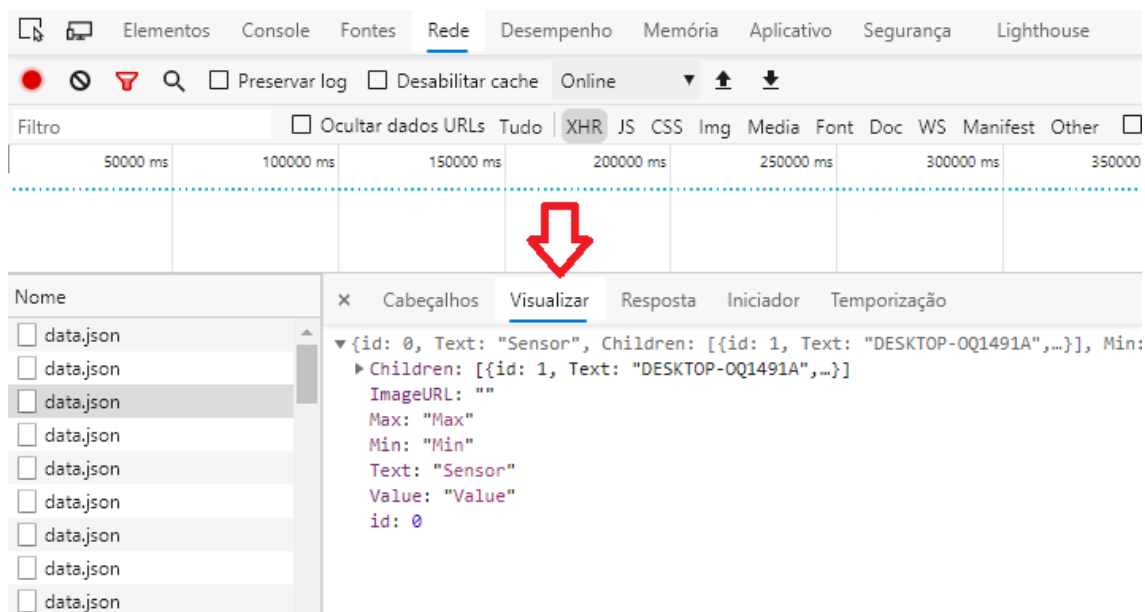
Olha só o nosso formato padrão. Vamos abrir um para ver a sua estrutura. Json

Já começam a aparecer as informações da captura dos dados:

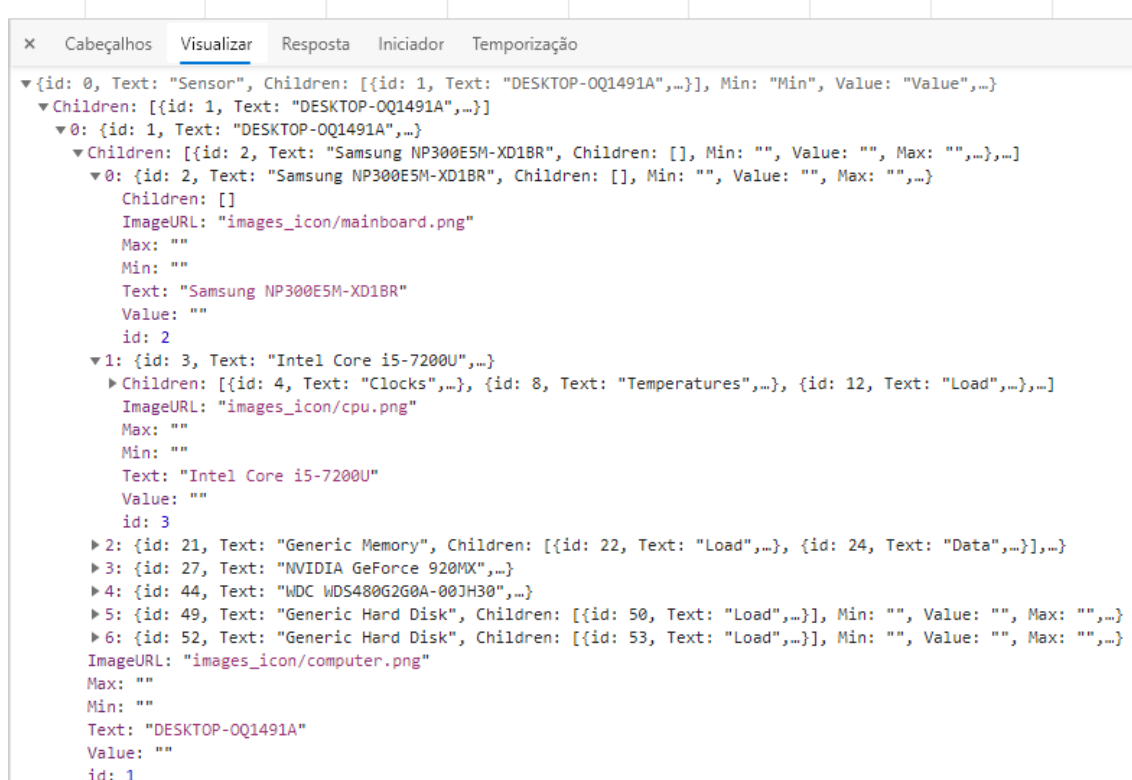


Clique em visualizar para obter uma formatação da estrutura mais inteligível.

API PYTHOAMS



Aqui vc já pode começar a comparar os códigos E A ESTRUTURA aqui e os do Pythohms



Em um mundo perfeito, todos os dados de que você precisa seriam apresentados de forma limpa em um formato aberto e bem documentado, que você poderia baixar facilmente e usar para qualquer propósito que precisar.

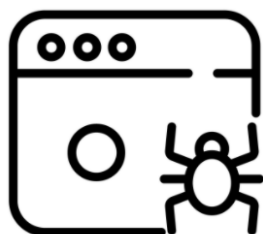
Freqüentemente, as informações de que você precisa ficam presas dentro de um site. Embora alguns sites façam um esforço para apresentar os dados em um formato de dados estruturados

API PYTHOHMS

e limpos, muitos não o fazem. Rastrear , extrair , processar e limpar dados é uma atividade necessária para uma série de atividades, desde o mapeamento da estrutura de um site até a coleta de dados em um formato exclusivo da Web ou talvez trancados em um banco de dados proprietário.

Mais cedo ou mais tarde, você descobrirá a necessidade de fazer alguns crawlers para obter os dados de que precisa e, quase com certeza, precisará codificar um pouco para que tudo seja feito da maneira certa. Depende de você como fazer isso, mas descobrirá que a comunidade Python é uma grande provedora de ferramentas, estruturas e documentação para obter dados de sites. No mundo real, os dados são confusos, raramente embalados como você precisa e, muitas vezes, desatualizados.

Crawlers são usados para tarefas de manutenção automatizadas em um Web Site, como checar os links ou validar o código HTML. Os Crawlers também podem ser usados para obter tipos específicos de informações das páginas da Web, como minerar endereços de email ou informações de repositórios.



Agora vamos rodar o crawler do OHM.

Baixe o Pythohms.py do Moodle

Criei uma pasta no seu desktop e carregue a Pythohms

Abra essa pasta pelo VSCode

Abra o código da Pythohms.py no ambiente de desenvolvimento

```
PS C:\Users\Marise\Desktop\CRAWLER Pythohms> ls
```

Mode	LastWriteTime	Length	Name
-a----	08/10/2020 14:21	3642	crawlerOpenHardwareMonitor.py

ATENÇÃO : o nome do meu crawler é diferente do disponibilizado no moodle. Nada muda. Aqui mostro que criei uma pasta e coloquei a API do moodle dentro da pasta.

Agora execute os comandos a seguir:

API PYTHOHMS

PS C:\Users\Marise\Desktop\CRAWLER Pythohms> python3 -m venv env

PS C:\Users\Marise\Desktop\CRAWLER Pythohms> .env\Scripts\activate.ps1

(env) PS C:\Users\Marise\Desktop\CRAWLER Pythohms> ls

Diretório: C:\Users\Marise\Desktop\CRAWLER Pythohms

Mode	LastWriteTime	Length	Name
----	-----	-----	----
d-----	08/10/2020 22:48		env

(env) PS C:\Users\Marise\Desktop\CRAWLER Pythohms> python .\crawlerOpenHardwareMoni

Observe que pode erro de requests, então instale a seguir:

(env) PS C:\Users\Marise\Desktop\CRAWLER Pythohms> pip install requests

Ok, vamos rodar a API agora. Antes mantenha o seu Open Hardware Monitor via servidor remoto web ativo em 2 segundos; Observe o endereço e a porta no código da API Pythohms e faça o mesmo no servidor Open hardware monitor.

Agora rode a API Pythohms

(env) PS C:\Users\Marise\Desktop\CRAWLER Pythohms> python .\crawlerOpenHardwareMoni

Resultado final da API Pythohms:

```
(env) PS C:\Users\Marise\Desktop\CRAWLER Pythohms> python .\crawlerOpenHardwareMoni
or.py920MX', 'WDC WDS480G2G0A-00JH30']}]
{'Desktop': 'DESKTOP-OQ1491A', 'MotherBoard': 'Samsung NP300E5M-XD1BR', 'CPU': [{'Name': 'Core 1', 'Clock': '3113,8 MHz', 'Temperature': '58,0 °C', 'Load': '25,8 %'}, {'Name': 'Core 2', 'Clock': '3113,8 MHz', 'Temperature': '60,0 °C', 'Load': '23,4 %'}], 'Memory': {'Load': '85,4 %', 'Use': '6,8 GB', 'Available': '1,2 GB'}, 'VideoCard': None, 'AllDevices': ['Samsung NP300E5M-XD1BR', 'Intel Core i5-7200U', 'Generic Memory', 'NVIDIA GeForce 920MX', 'WDC WDS480G2G0A-00JH30']}python .\crawlerOpenHardwareMoni
{'Desktop': 'DESKTOP-OQ1491A', 'MotherBoard': 'Samsung NP300E5M-XD1BR', 'CPU': [{'Name': 'Core 1', 'Clock': '3113,8 MHz', 'Temperature': '48,0 °C', 'Load': '10,9 %'}, {'Name': 'Core 2', 'Clock': '3113,8 MHz', 'Temperature': '49,0 °C', 'Load': '10,9 %'}], 'Memory': {'Load': '77,3 %', 'Use': '6,1 GB', 'Available': '1,8 GB'}, 'VideoCard': None, 'AllDevices': ['Samsung NP300E5M-XD1BR', 'Intel Core i5-7200U', 'Generic Memory', 'NVIDIA GeForce 920MX', 'WDC WDS480G2G0A-00JH30']}
(env) PS C:\Users\Marise\Desktop\CRAWLER Pythohms> python .\crawlerOpenHardwareMonitor.py
{'Desktop': 'DESKTOP-OQ1491A', 'MotherBoard': 'Samsung NP300E5M-XD1BR', 'CPU': [{'Name': 'Core 1', 'Clock': '1808,0 MHz', 'Temperature': '48,0 °C', 'Load': '7,8 %'}], 'Memory': {'Load': '77,3 %', 'Use': '6,1 GB', 'Available': '1,8 GB'}, 'VideoCard': None, 'AllDevices': ['Samsung NP300E5M-XD1BR', 'Intel Core i5-7200U', 'Generic Memory', 'NVIDIA GeForce 920MX', 'WDC WDS480G2G0A-00JH30']}
```

Desafios:

Rodar esta API em SO Windows com o OpenHardware monitor for Windows

Automatizar com loop sem interrupção.

Ajustar de dados de interesse, incluir ou excluir da API.

API PYTHOAMS

ATENÇÃO

AVALIAÇÃO DIAGNÓSTICA 2 TIPO PROJETO:

DATAS: PRÉVIAS EM 16/10 FINAL: 23/10 FORMATO PRESENCIAL (AJUSTES)

Ajustar a **estrutura de dados** para conexão com o banco

Criar a tabela no banco compatível com a **estrutura necessária ao projeto**.

ATENÇÃO AS MÉTRICAS E MEDIDAS, **devem ser fidedignas as reais**.

Sincronizar os inserts das tabelas **em 5 segundos**. Pode ajustar **conforme as médias e quartis** devem ser representados (frequência de atualização dos dados)

Fazer a conexão com banco local

Esta API deve está funcionando como a API Pythonics: no SO Windows e Linux (VM Local). Apresentar as duas APIs rodando nos dois ambientes de SO.

NÃO USE NESTE MOMENTO BANCO EM NUVEM. O PROJETO PREVÊ ARMAZENAMENTO LOCAL.

Importar as duas tabelas populadas no Banco pelas duas APIs, **no Rstudio** e realizar a **modelagem matemática**, aplicando **alguns métodos estudados** e **de visualização para melhor compreensão dos conjuntos de dados**.

Desafio extra: associar a variável de temperatura externa como fator de influência no desempenho da CPU. Verifique dados do site do **Climatempo**, **histórico de 30 dias anteriores** e **previsão de 15 dias à frente (2 situações distintas)**. Se a temperatura do clima tempo é diária, como você poderá comparar com os seus dados que estão sendo lidos a cada 2 segundos. **Verifique como transformar os dados lidos pelas APIs, em medida com mensuração diária.** Lembre-se onde você fará isso???